DISEÑO E IMPLEMENTACIÓN DE UNA RED PARA CONTROL Y MONITOREO REMOTO DE UNA PLANTA DE NIVEL

EDUARDO A. VELÁSQUEZ GONZÁLEZ

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
UNIDAD ACADÉMICA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2003

DISEÑO E IMPLEMENTACIÓN DE UNA RED PARA CONTROL Y MONITOREO REMOTO DE UNA PLANTA DE NIVEL

EDUARDO A. VELÁSQUEZ GONZÁLEZ

Proyecto de Grado para optar por el titulo de Ingeniero Electrónico

Director

DIEGO MARTÍNEZ

Jefe del Área de Electrónica Digital

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
UNIDAD ACADÉMICA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2003

Nota de aceptación:

Trabajo aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar por el titulo de Ingeniero Electrónico.

IVAN HERRERA_____ __
Firma de jurado

HECTOR FAVIO ROJAS__
Firma del jurado

Santiago de Cali 16 de Diciembre del 2003

AGRADECIMIENTOS

DIOS mío señor **JESUCRISTO** a ti la gloria y el honor por siempre, pues tú eres quien me ha dado todo y por eso debo mi esfuerzo y dedicación a ti **DIOS** de la historia, del presente y del futuro. Gracias infinitas a ti creador y salvador del mundo. **AMEN**

A MARIA SANTÍSIMA, madre del cielo que intercede por nosotros sus hijos, ante nuestro señor **JESUCRISTO**.

A mi esposa **ADRIANA LENIS** que me ha dado el apoyo necesario para mantenerme firme en la culminación de este proyecto, que por la gracia de **DIOS** podamos permanecer juntos siempre y que DIOS nos bendiga con una familia que **AME**.

A mi mamá **ESPERANZA GONZÁLEZ** que descansa en la paz del señor y que intercede desde el cielo por mí, a ella gracias por haberme enseñado a valorar la vida.

A mi hermana **LUZ ÁNGELA**, mi hermano **ANDRÉS** y mi papa **DONATO**, quienes son con los que he compartido mi vida desde mi infancia, en una gran unidad familiar, gracias por ellos a **DIOS**.

A todos mis familiares y amigos con los que también he compartido y han sido de gran ayuda en mi vida, que no escribo sus nombres porque no me cabrían aquí, pero que siempre los llevo en mi corazón, también gracias a **DIOS** por ellos.

A JORGE LENIS y su esposa **OMAIRA**, quienes han sido gran apoyo y antorcha en el camino de la vida hacia **DIOS** nuestro padre, a ellos les debo mucho de mi conversión y mi crecimiento en la fe católica.

A DIEGO MARTÍNEZ, gran amigo y director de este proyecto. Que **DIOS** lo bendiga siempre en su camino.

A LUIS H. PÉREZ, por su gran labor como rector de la universidad, que **DIOS** le siga dando mucha luz para seguir dirigiendo este rebaño de la **UAO**.

A los directivos, profesores y demás trabajadores de la **UAO**, porque todos han sido participes en la construcción de un mundo mejor, a través de su labor. **El SEÑOR** derrame mil bendiciones sobre todos ellos.

Y a la **SANTA IGLESIA** de **DIOS**, dirigida por nuestro sumo pontífice, **JUAN PABLO II**, quien se ha esmerado por llevar la paz a todos los rincones del mundo. Gracias **IGLESIA** mía por permanecer unida en el amor, en la paz y en la verdad.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

Este proyecto propone el diseño e implementación de un sistema de control de nivel en red, usando una red de microcontroladores, con protocolo CAN, el cual tiene gran auge en mercado industrial en los últimos años. También posee ciertas ventajas que lo hacen efectivo para las tareas que requieran tiempo real.

En el diseño se construyen tres módulos que conforman el sistemas; el modulo de medición de nivel que se encarga de medir el nivel en un tanque de 25cm de profundidad por 20 de diámetro y a su vez envía esta información por la red CAN con los chip que se mencionan en seguida. En el segundo modulo se hace el control de una válvula de ½" con un motor paso a paso por medio de un sistema de control difuso implementado en ensamblador para el microcontrolador de Motorola MC68HC08JK3. Este micro también se encarga de manipular un conjunto de registros en el MCP2510 que es un chip tiene implementado la capa de enlace de datos del protocolo CAN, en donde el nivel físico es manejado por el driver UC5350 de la Texas, quienes transportan la información de apertura y de nivel entre los módulos. Por ultimo encontramos el diseño y construcción de una tarjeta tipo ISA para PC que pose chips de comunicaciones similares y algunos componentes adicionales unidos al software desarrollado en Visual Estudio 6, para hacer el monitoreo y configuración de algunos parámetros de la planta.

Cada modulo mencionado visualiza los datos que están manipulando usando una pantalla de LCD, para el caso de Medidor ultrasónico para mostrar el nivel y la Válvula inteligente o difusa para mostrar en nivel deseado y la apertura de la válvula. En el modulo ISA la visualización de los datos se hace en la computadora en la interface usuario.

INTRODUCCIÓN

Actualmente las comunicaciones juegan un papel importante en la industria, ya que con las nuevas herramientas tecnológicas se hace posible implementar redes de procesadores digitales de bajos costos y buen desempeño. Capaces de intervenir en procesos industriales, con el fin de llevar información de un nodo a otro para realizar tareas de control y/o monitoreo y gestión de procesos en tiempo real.

Los sistemas automáticos de control modernos tienden entonces a convertirse en un conjunto de sistemas distribuidos que tienen objetivos individuales dentro de un proceso, así mismo entregarán o recibirán información a través de un bus de red donde intervienen otros dispositivos que se encuentren en él. Todo un conjunto de procesos puede ser monitoreado desde cualquier lugar siendo posible acceder directamente a cada dispositivo ubicado en la red, y de acuerdo a su función en el proceso podrá ser configurado o monitoreado.

Es por estas razones que se ha considerado muy importante para nuestra institución el desarrollo de un proyecto en el cual se estudie e implemente un protocolo estándar de comunicaciones industriales que ha tenido mucho éxito actualmente, como es el caso de CAN.

1 PLANTEAMIENTO

Los sistemas de control distribuido utilizados en la industria deben resolver dos asuntos, el primero es que los dispositivos utilizados para implementar el canal de comunicaciones han de ofrecer un canal confiable proporcionando una aceptable inmunidad al ruido, puesto que en las plantas de procesos industriales existen condiciones de trabajo que son adversas, tales como el ruido producido por los contactos y los motores de inducción magnética. El segundo aspecto a considerar es la capacidad que tengan estos dispositivos para entregar información oportuna a los encargados de verificar los diferentes procesos, esto está ligado con las características que implementa el protocolo de comunicaciones para que los diferentes módulos conectados a la red puedan realizar transferencias de información en instantes predecibles, de tal forma que el desempeño de los algoritmos de control sea adecuado.

1.1 FORMULACIÓN DEL PROBLEMA

¿Cómo desarrollar un sistema de control de nivel con una red de Microcontroladores que cumpla con las especificaciones necesarias para su correcto funcionamiento de los algoritmos de control, y permita una adecuado monitoreo del proceso y configuración de parámetros del sistema?

2 MARCO TEÓRICO

En aplicaciones industriales de control distribuido existen muchas razones para utilizar sistemas de bus. Algunas ventajas de estos sistemas son: 1) El bus de campo reemplaza equipos de cable delgado por cable de par trenzado o fibra óptica, lo cual representa considerables ahorros en términos de cableado y en el coste de las instalaciones, además la función directora de la red asociada permite rápidos diagnósticos y resoluciones de problemas de los equipos conectados. 2) El uso de buses de campo permite que los equipos puedan ser operados remotamente vía red, simplificando gradualmente la calibración y configuración del trabajo que se requiere en cada momento. 3) Los nuevos equipos se integran también más fácilmente. 4) Otro punto más es la estructura modular de los buses de campo, que permite construir sistemas de control desde equipos estandarizados, lo que es más, equipos inteligentes, como controladores de programa de abastecimiento/almacenamiento (SPC), pueden conectarse vía bus de campo, permitiendo procesar la información y distribuirla a lo largo de varios equipos. 5) Una ventaja final es la intercambiabilidad que, desde la estandarización de equipos de buses de campo, significa el reemplazar los equipos en el campo, configuración, almacenamiento y mantenimiento.

Todo esto son las ventajas de los sistemas de bus. El único problema es el "Sí, pero..." que viene después de todos estos puntos listados. Pongamos en contexto el "Ahorrar en costes de instalación y cableado": Necesitamos pararnos y considerar que todo el conjunto de cableado representa menos del 10 % del valor de los sistemas de comunicación. Además, el número de puntos de terminación está aumentando y las conexiones, cajas de unión, conectores de bus y protección, son más caras y por tanto se intensifica el costo.

La situación es similar con respecto a "fácil de ampliar y recolocar". Para ser capaces de integrar fácilmente nuevos equipos en sistemas ya existentes, los protocolos deben

permanecer estables por muchos años y éste no es el caso, incluso en los protocolos estandarizados.

A pesar de los problemas indicados arriba, está claro que no hay alternativas a los sistemas de bus. Consecuentemente los sistemas de bus individuales, ya han sido aceptados en mayor o menor medida en algunos sectores. Por ejemplo, el CAN bus se ha establecido muy bien en la Ingeniería de Automoción y ha ido incrementando su popularidad en la Tecnología de Automoción. El INTERBUS-S, el Bitbus y el Profibus se encuentran frecuentemente en entornos de Automatización Industrial.

Las configuraciones de planta distribuida han ganado aceptación en la tecnología de automatización, esto a contribuido a reducir costos de cableado, mientras se incrementa considerablemente la flexibilidad y rentabilidad de la planta.

3 OBJETIVOS

3.1 GENERALES

- Implementar un sistema de control de nivel, utilizando una red de Microcontroladores.

- Desarrollar el software de usuario para monitoreo y configuración remota del sistema de control de nivel.

3.2 ESPECÍFICOS

- Diseñar y construir una válvula de rotación mecánica que es manipulada por un motor paso a paso. La válvula tendrá la capacidad de comunicarse de forma serial con un estándar de comunicaciones para aplicaciones Industriales, además llevara implementado un controlador difuso.

- Diseñar y construir un sistema de medición de nivel que tenga capacidad de comunicarse por medio de un estándar para aplicaciones Industriales.

- Diseñar y construir un sistema de comunicación con un estándar para aplicaciones Industriales, que permita hacer la interfase con el computador y el sistema de control implementado.

- Diseñar e implementar el software de comunicación para hacer el control y monitoreo de la planta en forma remota.

4 JUSTIFICACIÓN

Este proyecto da un paso para el trabajo con sistemas distribuidos, los cuales hoy en día dejan ver un panorama prometedor en cuanto a las aplicaciones de control, en las cuales se puede hacer configuración y monitoreo remoto de cualquier sistema de control que tenga este tipo de implementación. Entonces se tendrá una base de partida para continuar con trabajos de este tipo, con lo cual en un futuro se podrá tener una red con todas las plantas que existen en el laboratorio de control de la universidad, a la cual se accederá de manera remota con el fin de observar el comportamiento de cada planta y de igual manera se tenga la posibilidad de configurar sus diferentes parámetros.

Este trabajo puede abrir paso a desarrollos en este campo de control, en el cual se utilizan los sistemas de comunicación para interactuar con los elementos que conforman un proceso industrial, esto puede llegar a beneficiar la industrial colombiana, ya que los nuevos desarrollos en este campo pueden ser llevados a la práctica en la industria.

5 PLANTEAMIENTO DEL DISEÑO DEL SISTEMA

Para el desarrollo del sistema de control hay que tener en cuanta una serie de cuestiones que deben ser evaluadas, tales como, el sistema de comunicaciones de bus que se utilizará en el proyecto, por ello en el capitulo 6, podemos ver la explicaciones de los sistemas de bus existentes en el mercado de los cuales se escogió CAN por las razones que se explican en las conclusiones del capitulo 7, donde vemos ampliamente las ventajas de CAN sobre las otras tendencias de redes Industriales.

Teniendo claro que se escoge CAN para la realización del diseño, es necesario hacer una serie de investigaciones en cuanto a chips que tengan facilidad para su implementación y su consecución en el mercado, para nuestro proyecto se consiguieron los MCP2510 de la Microchip que resulta ser un chip de bajo costo y fácil manejo, a través de una interface SPI, que permite la lectura y escritura de los registros del chip, para así poder hacer las configuraciones y comunicaciones entre los nodos participantes de la red CAN (Capitulo 8 Sección 4).

Bueno pero, ¿Quiénes se encargaran de manipular la información que se tramita en la red? Para ello se eligió el microcontrolador de la Motorola MC68HC908JK3, pues este tiene características de bajo costo, fácil manejo (Set de Instrucciones Reducido pero muy robusto), montaje de tarjeta de desarrollo con pocos elementos de fácil consecución en el mercado nacional y software de desarrollo gratuito, en resumidas cuentas a la hora de tener un diseño final es muy conveniente este tipo de microcontroladores, el cual es tratado en la sección 2 del capitulo 8. También se debe hacer la interacción de la red CAN y la computadora, por lo que se opto por el puerto ISA, pues es un puerto del computador que tiene acceso rápido a los datos y aun existen computadores con este puerto. Otra ventaja es que el diseño de una tarjeta ISA es sencillo y se requiere de pocos elementos para su construcción. Esto lo vemos en la sección 1 del capitulo 8.

Ahora hay que medir nivel, controlar una válvula, hacer un programa que haga el monitoreo del sistema. ¿Esto de diseñar tiene su cuento No? así es la electrónica, hasta mecánica hay que trabajar. Estas cuestiones de diseño de un medidor de Ultrasonido, una Válvula Inteligente (control Difuso) se pueden ver en el capitulo 8 secciones 5.1 y 5.2. Y en cuanto a la interface usuario, será necesario diseñar un programa en Visual Basic, pues es una herramienta de desarrollo rápido y de fácil acceso a los estudiantes de la Universidad Autónoma, esto se explica en el Capitulo 8 Sección 7.

6 ASPECTOS BÁSICOS DE LAS REDES DE COMUNICACIÓN PARA EL DESARROLLO DE APLICACIONES DE TIPO INDUSTRIAL

Las comunicaciones entre los instrumentos de proceso y el sistema de control se basan principalmente en señales analógicas (neumáticas de 3 a 15 psi en las válvulas de control y electrónicas de 4 a 20 mA cc). Pero ya existen instrumentos digitales capaces de manejar gran cantidad de datos y guardarlos históricamente; su precisión es diez veces mayor que la de la señal típica de 4-20 mA cc. En vez de transmitir cada variable por un par de hilos, transmiten secuencialmente las variables por medio de un cable de comunicaciones llamado bus. La tecnología fieldbus (bus de campo) es un protocolo de comunicaciones digital de alta velocidad que esta creada para remplazar la clásica señal de 4-20 mA que aún se utiliza en muchos de los sistemas DCS (Sistema de Control Distribuido) y PLC (Controladores Lógicos Programables), instrumentos de medida y transmisión y válvulas de control. La arquitectura fieldbus conecta estos instrumentos con computadores que se usan en diferentes niveles de coordinación y dirección de la planta. Muchos de los protocolos patentados para dichas aplicaciones tiene una limitante y es que el fabricante no permite al usuario final la interoperabilidad de instrumentos, es decir, no es posible intercambiar los instrumentos de un fabricante por otro similar. Es claro que estas tecnologías cerradas tienden a desparecer ya que actualmente es necesaria la interoperabilidad de sistemas y aparatos y así tener la capacidad de manejar sistemas abiertos y estandarizados. Con el mejoramiento de los protocolos de comunicación es ahora posible reducir el tiempo necesitado para la transferencia de datos, asegurar la misma, garantizar el tiempo de sincronización y el tiempo real de respuesta determinística en algunas aplicaciones.

6.1 LA TECNOLOGÍA DE BUSES DE CAMPO

Físicamente podemos considerar a un *bus* como un conjunto de conductores conectando conjuntamente más circuitos para permitir el intercambio de datos. Contrario a una

conexión punto a punto donde solo dos dispositivos intercambian información, un bus consta normalmente de un número de usuarios superior, además que generalmente un bus transmite datos en modo serial, a excepción de algún protocolo de bus particular como SCSI, o IEEE-488 utilizado para interconexión de instrumentos de medición, que no es el caso de los buses tratados como buses de campo. Para una transmisión serial es suficiente un número de cables muy limitado, generalmente son suficientes dos o tres conductores y la debida protección contra las perturbaciones externas para permitir su tendido en ambientes de ruido industrial.

6.1.1 Ventajas de un bus de campo:

- El intercambio se lleva a cabo por medio de un mecanismo estándar.
- Flexibilidad de extensión.
- Conexión de módulos diferentes en una misma línea.
- Posibilidad de conexión de dispositivos de diferentes procedencias.
- Distancias operativas superiores al cableado tradicional.
- Reducción masiva de cables y costo asociado.
- Simplificación de la puesta en servicio.

6.1.2 Desventajas de un bus de campo:

- Necesidad de conocimientos superiores.
- Inversión de instrumentación y accesorios de diagnóstico.
- Costos globales inicialmente superiores.

6.1.3 Procesos de comunicación por medio de bus

El modo más sencillo de comunicación con el bus es el sondeo cliente/servidor. Más eficiente pero también más costoso es el Token bus (IEEE 802.4), desde el punto de vista

físico tenemos un bus lineal, desde el punto de vista lógico un token ring. El procedimiento token passing es una combinación entre cliente/servidor y token bus. Todo servidor inteligente puede ser en algún momento servidor (Ej. PROFIBUS Si el bus se cierra formando un anillo, obtenemos un token ring (IEEE 802.5)

6.1.4 Algunos Tipos De Bus

La mayoría de los buses trabajan en el nivel 1 con interfaz RS 485.

6.1.4.1 ASI (Actuator Sensor Interface)

Es el bus más inmediato en el nivel de campo y más sencillo de controlar, consiste en un bus cliente/servidor con un máximo de 31 participantes, transmite por paquetes de solo 4 bits de datos. Es muy veloz, con un ciclo de 5 ms aproximadamente. Alcanza distancias de máximo 100 m.

6.1.4.2 Bitbus

Es el más difundido en todo el mundo, es cliente/servidor que admite como máximo 56 clientes, el paquete puede transmitir hasta 43 bytes de dato.

6.1.4.3 Profibus (PROcess FIeld BUS)

Es el estándar europeo en tecnología de buses, se encuentra jerárquicamente por encima de ASI y BITBUS, trabaja según procedimiento híbrido token passing, dispone de 31 participantes hasta un máximo de 127. Su paquete puede transmitir un máximo de 246 Bytes, y el ciclo para 31 participantes es de aproximadamente 90 ms. Alcanza una distancia de hasta 1200 m.

6.1.4.4 DP estándar DIN E 19245 T3

Para aplicaciones de bajo costo en redes sensor/actuador. Es especial para transmisión de mensajes cortos transferidos a alta velocidad, para 32 nodos y 200 m de red a 1,5 Mbps, que resulta en un intercambio de 1000 bit de datos de sensor/actuador en menos de 10 ms.

6.1.4.5 El Bus CAN

En este protocolo esta definido el nivel de enlace de datos y el nivel físico en la arquitectura OSI. Transmisión de mensajes cortos con resolución de contienda, consistencia de información en los nodos con velocidades máximas de 1Mbit/s, y 40 metros de cable y 500Kbits, para distancias de hasta 1000 metros, intervención de hasta 30 nodos

6.2 CLASIFICACIÓN DE LAS REDES INDUSTRIALES.

Si se clasifican las redes industriales en diferentes categorías basándose en la funcionalidad, se hará en:

6.2.1 Buses Actuadores y Sensores

Inicialmente es usado un sensor y un bus actuador en conexión simple, dispositivos discretos con inteligencia limitada, como un foto sensor, un switch limitador o una válvula solenoide, controladores y consolas terminales. Los sensores y buses actuadores como ASI y CAN (Control Advanced Network), son diseñados para que el flujo de información sea reducido a pocos bits.

6.2.2 Buses de Campo y Dispositivos

Estas redes se distinguen por la forma como manejan el tamaño del mensaje y el tiempo de respuesta. En general estas redes conectan dispositivos inteligentes en una sola red distribuida. Estas redes ofrecen altos niveles de diagnóstico y capacidad de configuración, generalmente al nivel del poder de procesamiento de los dispositivos más inteligentes. Son las redes más sofisticadas que trabajan con control distribuido real entre dispositivos inteligentes, tal es el caso de FIELDBUS FOUNDATION. Comúnmente dichas redes incluyen en los dispositivos y buses de campo las clases CANOpen, DeviceNet, FIELDBus Foundation, Interbus-S, Lonwork, Profibus- DP y SDS.

6.2.3 Buses de Control

Típicamente los buses de control para redes de punto a punto entre controladores como PLC (Controlador Lógico Programable), DCS (Sistemas de control distribuido), Sistemas de consolas terminales usados para HMI (Interface Hombre Máquina), archivamiento histórico y control supervisor; son usados para coordinar y sincronizar el control entre las unidades de producción y las celdas de manufactura. Usualmente son empleados como buses controladores para redes industriales (Control Net, Profibus-FMS, Map) Adicionalmente, puede utilizarse redes Ethernet con protocolo TCP/IP como bus controlador para conectar dispositivos de alto nivel y consolas terminales. Pueden también usarse redes Ethernet como GATEWAY para conectar otras redes industriales. En este caso es recomendado aislar el segmento de la red industrial Ethernet del bus principal con Bridge para hacer el segmento independiente.

6.2.4 Componentes De Las Redes Industriales

En grandes redes industriales un simple cable no es suficiente para conectar el conjunto de todos los nodos de la red. Deben definirse topologías y diseños de redes para proveer un aislamiento y conocer los requerimientos de funcionamiento.

6.2.5 Repetidor

El repetidor o amplificador es un dispositivo que intensifica las señales eléctricas para que puedan viajar grandes distancias entre nodos. Con este dispositivo se pueden conectar un gran número de nodos a la red; además se pueden adaptar a diferentes medios físicos como cable coaxial o fibra óptica.

6.2.6 Enrutadores

Es un switch "Enrutador" de paquetes de información, este se encarga de direccionar los mismos entre diferentes segmentos de red que definen la ruta hacia el nodo destino.

6.2.7 Bridge

Se usan para conectar diferentes secciones de red, las cuales pueden tener diferentes características eléctricas y protocolos de acceso al medio.

6.2.8 Gateway

Un gateway es similar a un puente ya que suministra interoperabilidad entre buses y diferentes tipos de protocolos y además las aplicaciones pueden comunicarse a través de él. La diferencia es que las redes pueden variar incluso en los niveles superiores.

6.3 TOPOLOGÍA DE REDES INDUSTRIALES

Los sistemas industriales usualmente consisten de dos o más dispositivos interconectados a través de una red de comunicaciones.

Como un sistema industrial puede ser bastante grande, debe considerarse la topología de la red a implementar en una determinada aplicación. Las topologías más comunes son:

6.3.1 Red Bus

Enlaza todos los dispositivos en serie por conexiones extensas con un mismo cable; dependiendo del tipo de red muchos nodos pueden estar empalmados en el bus y comunicarse con otros nodos por el mismo cable. Además, esta topología es simple de entender y fácil de extender, pero también presenta una serie de desventajas, por ejemplo: una ruptura del cable puede causar fallas de comunicación a un número de dispositivos y la congestión debida al tráfico de la red reduce las posibilidades de comunicación en la misma.

En la actualidad este tipo de redes son las más utilizadas en aplicaciones de control industrial.

6.3.2 Red Estrella

Tiene un controlador central y uno o más segmentos de conexión de red que parten del concentrador. Con esta topología se pueden agregar fácilmente nuevos nodos sin interrumpir la operación de la red. Entre los beneficios de esta topología se encuentran que ante la falla de un dispositivo no se interrumpe la comunicación entre algunos otros dispositivos y la red; pero al fallar el concentrador la red entera falla.

6.3.3 Red Híbrida

Esta topología ha tenido mucha difusión en aplicaciones industriales, ya que permite la combinación de las topologías bus y estrella para crear grandes redes que consisten en

concertadores y miles de dispositivos iguales. Su configuración es muy popular en las redes industriales Ethernet, IELDBus Foundation, Device Net, Profibus y CAN, usando buses híbridos y topología estrella dependiendo de la aplicación requerida. Las redes en topología Híbrida ofrecen las ventajas y desventajas de las topologías de red Bus y Estrella, se puede configurar dicha red híbrida para que al fallar un dispositivo no se quede otro fuera de la red, y se pueden adicionar o retirar segmentos de red sin afectar algún nodo de la ya existente.

6.4 BENEFICIOS DE UNA RED INDUSTRIAL

- Reducción de cableado (físicamente) - Dispositivos inteligentes (funcionalidad y ejecución)
- Posibilitan el desarrollo de aplicaciones de control distribuido (Flexibilidad) .
- Simplificación de cableado de las nuevas instalaciones
- Reducción de costo en cableado y cajas de conexión
- Aplicable a todo tipo de sistema de manufactura.
- Incremento de la confiabilidad de los sistemas de producción
- Optimización de los procesos existentes.

6.5 REDES INDUSTRIALES CON PLC

Muchos sistemas están conformados por equipos de diferentes fabricantes y funcionan en diferentes niveles de automatización; además, a menudo se encuentran distanciados entre sí; pero sin embargo, se desea que trabajen en forma coordinada para un resultado satisfactorio del proceso. El objetivo principal es la comunicación totalmente integrada en el sistema. Al usuario, esto le reporta la máxima flexibilidad ya que también puede integrar sin problemas productos de otros fabricantes a través de las interfaces software estandarizadas. En los últimos años, las aplicaciones industriales basadas en comunicación digitales se han incrementado haciendo posible la conexión de sensores, actuadores y

equipos de control en una planta de procesamiento. De esta manera, la comunicación entre la sala de control y los instrumentos de campo se han convertido en realidad. La Comunicación digital debe integrar la información provista por los elementos de campo en el sistema de control de procesos.

6.6 SOLUCIONES CON ETHERNET

Aunque los buses de campo continuarán dominando las redes industriales, las soluciones basadas en Ethernet se están utilizando cada vez más en el sector de las tecnologías de automatización, donde las secuencias de procesos y producción son controladas por un modelo cliente/servidor con controladores, PLC y sistemas ERP (Planificación de los recursos de la empresa), teniendo acceso a cada sensor que se conecta a la red. La implementación de una red efectiva y segura también requiere el uso de conectores apropiados, disponibles en una amplia variedad y para soluciones muy flexibles. Ethernet industrial es todavía un estándar en desarrollo, sin embargo, una de sus características principales es que es muy dinámica.

Diferentes institutos, como la Asociación Ethernet Industrial o IOANAEuropa, se han asociado para guiar y liderar a fabricantes y usuarios internacionales para establecer Ethernet como un estándar en todo el entorno industrial. La finalidad es crear comunicaciones globales en todos los niveles, desde la automatización de la fábrica, vía la automatización de los procesos productivos, hasta la automatización de edificios. En principio, los medios de transmisión para Ethernet que hoy están disponibles son, por una parte y como solución más habitual, el cable coaxial para 10Mbits/seg, el cable Thin-Ethernet-BNC para 10Base2 y el cable thick-Ethernet para 10Base5. Por otra parte, para 10Mbit/seg o 100 Mbits/seg (4 líneas de cable trenzado, tanto apantallado como sin apantallar) se utilizan cables de par trenzado. Una efectiva solución para realizar una aplicación de control de procesos industriales es utilizar tecnología de *gateway* con el cual podemos utilizar la red corporativa (sea esta thernet, ATM, WLAN, FDDI o Token Ring) e integrar dicha red al monitoreo, supervisión, control y aplicaciones de adquisición de datos.

De esta forma podemos hacer control desde estaciones de monitoreo (que pueden ser remotas si existe la necesidad) conectadas directamente a la red TCP/IP de la empresa.

6.7 CONCLUSIONES

- Cada vez se esta acabando con tecnologías cerradas; que en un mundo en proceso de globalización, es imposible que sobrevivan.
- A nivel industrial se esta dando un gran cambio, ya que no solo se pretende trabajar con la especificidad de la instrumentación y el control automático, sino que existe la necesidad de mantener históricamente información de todos los procesos, además que esta información este también en tiempo real y que sirva para la toma de decisiones y se pueda así mejorar la calidad de los procesos.
- Las condiciones extremas a nivel industrial requieren de equipos capaces de soportar altas temperaturas, ruido excesivo, polvo, humedad y demás condiciones adversas; pero además requiere de personal capaz de ver globalmente el sistema de control y automatización industrial junto con el sistema de red digital de datos.

7 RED DE CONTROLADORES CAN

De acuerdo a las últimas tendencias de las comunicaciones y al gran auge de los sistemas microcontrolados distribuidos en las distintas aplicaciones de la industria se ha optado por escoger para este proyecto una red con esquema de bus tipo CAN. Las ventajas de utilizar este protocolo es que debido a que es un protocolo de fácil implementación actualmente, bien sea a través del uso de ASICs o microcontroladores que lo traen implementado, tienen una alta difusión en las actuales aplicaciones de tipo industrial. También ofrece muy buenas características para el desarrollo de aplicaciones de tiempo real y sus métodos para detección y corrección de fallas lo hacen adecuado para la implementación en sistemas críticos.

Los controladores CAN y las interfaces son físicamente pequeños, tienen bajo costo, componentes ya desarrollados, estos operan a una alta velocidad para tiempo real y bajo condiciones adversas. Todas estas propiedades le permiten al CAN estar no solo en la industria de autos sino también en una gran variedad de aplicaciones.

Los beneficios como la reducción de costo y la fácil implementación que gano la industria de los carros, ahora están disponibles para un gran rango de productos de distintos fabricantes.

Por ejemplo:

- Control fluvial y sistema de navegación.
- Maquinaria para agricultura.
- Sistemas de control para producción en línea.
- Fotocopiadoras.
- Sistemas médicos.
- Telescopios ópticos.

- Maquinaria para producción de textiles.
- Maquinaria de empacamiento.
- Juguetes para niños.

También se usa el CAN para controlar redes de sensores, actuadores y transductores, todos los fabricantes de los productos arriba mencionados obtienen los siguientes beneficios:

- Reducción en el tiempo de diseño (componentes multi-fuentes, tareas)
- Bajo costo de conexión (ligeros, pequeños cables de conectores)
- Facilita la implementación (pocas conexiones).

Enseguida se hace una breve explicación a este protocolo.

CAN (Controller Area Network), fue inicialmente creado por el proveedor alemán de sistemas para automóviles Robert Bosh a mediados de los 80’s, para aplicaciones en automóviles como un método para permitir comunicaciones seriales robustas. El propósito era hacer automóviles más confiables, seguros y eficientes disminuyendo la dura y compleja tarea del cableado. Partiendo de este principio, el protocolo CAN ha ganado amplia popularidad en la automatización industrial y en las aplicaciones automotrices. Otros mercados donde las soluciones de trabajo en red pueden brindar beneficios atractivos como en equipos médicos, equipos de prueba y maquinas móviles también comienzan a utilizar los beneficios de CAN.

El objetivo de este capitulo es explicar algunos conceptos básicos a cerca de CAN y mostrar los beneficios de escoger este para sistemas embebidos para aplicaciones de trabajo en red.

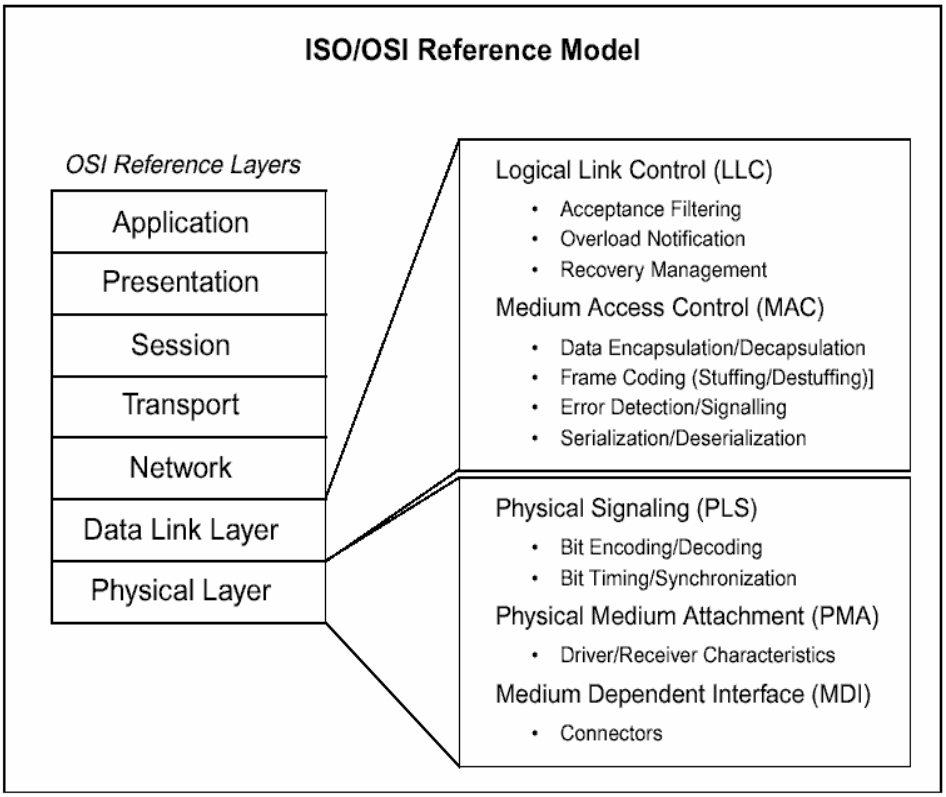
7.1 GENERALIDADES DEL CAN

El bus CAN *(Controller Area Networ*k) es un protocolo de comunicaciones serie, radiado y sensible a portadora, el cual permite la transmisión y recepción de mensajes pequeños en distancias cortas y ambientes ruidosos. Este puede funcionar a una velocidad máxima de un 1Mbit/s. El protocolo CAN hace resolución de la contención basado en la prioridad de los mensajes, Incorporando mecanismos para detectar y tolerar fallos de transmisión. Todos los nodos en el bus reciben los mismos datos, a esto se le llama consistencia de datos. Por ser una tecnología barata y eficiente tiene un amplio espectro de utilización como ya se menciono.

Muchas aplicaciones de red siguen un esquema de capas para la implementación de sistemas. Este esquema sistemático permite la interoperabilidad entre productos de diferentes fabricantes. Fue creado entonces un estándar por la Organización Internacional de Estándares (ISO) como una plantilla para el esquema de capas. Este es llamado modelo de referencia OSI (Open System Interconnection - **Sistema de Interconexión Abierto**) y se muestra en la Figura No.1. El protocolo CAN tiene implementadas las dos capas mas bajas del modelo de referencia OSI. El medio de comunicación fue intencionalmente omitido por las especificaciones del CAN de Bosch para permitir a los diseñadores adaptar y optimizar el protocolo de comunicaciones en múltiples medios, ofreciendo mayor flexibilidad (Par trenzado, Un solo cable, Aislamiento Óptico, RF, IR, etc.). A pesar de esta flexibilidad, se da la interoperabilidad.

Para facilitar algunos de estos conceptos, la Organización Internacional de Estándares y la Sociedad de Ingenieros Automovilísticos (SAE) definió protocolos basados en CAN que incluyen las especificaciones de las dos capas de bajo nivel. El ISO11898 es el estándar para aplicaciones de alta velocidad, ISO11519 es el estándar para aplicaciones de baja velocidad, y J1939 (de la SAE) es para aplicaciones en camiones y buses, estos tres protocolos especifican un bus eléctrico diferencial de 5V como interfase física.

Figura 1. Modelo de Referencia OSI de la ISO.



El resto de capas del protocolo ISO/OSI es dejada en manos de los desarrolladores de software. HLPs (Higher Layer Protocols- Protocolos de Nivel Superior) son generalmente usados para implementar las cinco capas superiores del modelo de referencia OSI. Las HLPs son usadas para:

- Procedimientos de Inicialización estándar incluyendo las ratas de bit usadas.
- Direcciones distribuidas entre nodos participantes o tipos de mensajes.
- Determinan la estructura del mensaje.
- Proveen rutinas de tratamiento de errores del sistema.
- Control de flujo de datos.
- Fragmentación de datos en paquetes de 8 bytes (en caso de ser necesario).
- Reconstrucción de los datos fragmentados.
- Organización y asignación de los mensajes, teniendo en cuenta las prioridades.

7.1.1 ARQUITECTURA EN NIVELES DE CAN

Subnivel de control del enlace lógico (LLC)

El subnivel LLC realiza las siguientes **funcione**s:

- Filtrado de los mensajes recibidos.
- Recuperación ante fallos de transmisión.
- Retransmisión automática de mensajes.

Los **servicios** ofrecidos son:

- Transmisión de mensajes.
- Solicitud remota de datos.
- Operaciones para la gestión de la conexión.

7.1.2 Servicios del nivel LLC

- Se proporcionan servicios de comunicación sin conexión y sin confirmación.
- La transferencia de datos puede ser punto a punto, multipunto y radiado.
- Una operación iniciada en el LLC es única, autocontenida e independiente.
- Los mensajes no incluyen destinatario.
- El identificador describe el significado del contenido del mensaje.
- Permite al usuario reiniciar la operación del nodo y conocer su estado operativo.

7.1.3 Control de acceso al medio (MAC)

Proporciona al LLC los siguientes **servicios:**

- Transmisión de mensajes.
- Solicitud remota de datos.

- Operaciones para la gestión de la conexión.
- Se proporcionan servicios de comunicación sin conexión y con confirmación.

El subnivel MAC realiza las siguientes **funcione**s:

- Encapsulado/desencapsulado de los mensajes enviados/recibidos
- Gestión del acceso al medio.
- Detección y notificación de errores de transmisión.
- Lógicamente compuesto por dos partes independientes (transmisión y recepción).

7.2 MODELO FUNCIONAL

7.2.1 Funciones de Transmisión

- Recepción de la información del LLC.
- Construcción del mensaje. Se añaden los campos necesarios y se calcula el CRC (Figura No.3).
- Se inicia el proceso de transmisión cuando el bus está ocioso. Se emplean las primitivas del nivel físico.
- Se arbitra el bus y se pasa a modo de recepción si se pierde el arbitraje.
- Detección de errores.
- Transmisión del mensaje de error.

7.2.2 Funciones de Recepción

- Recepción de bits del nivel físico.
- Regeneración de la estructura del mensaje.
- Detección de errores.
- Transmisión de confirmación.

- Transmisión del mensaje de error.
- Borrado de la información adicional del mensaje.
- Presentación del mensaje al subnivel LLC.

7.3 NIVEL FÍSICO

La capa física en CAN es responsable de la transferencia de bits entre los distintos nodos que componen la red. Define aspectos como niveles de señal, codificación, sincronización y tiempos en que los bits se transfieren al bus. En la especificación original de CAN, la capa física no fue definida, permitiendo diferentes opciones para la elección del medio y niveles eléctricos de transmisión. Las características de las señales eléctricas en el bus fueron establecidas más tarde por el estándar ISO 11898.

La especificación CiA (CAN in AUTOMATION, http://www.can-cia.de), complementó las definiciones respecto al medio físico y conectores. Los nodos conectados al bus interpretan dos niveles lógicos denominados:

- Dominante: la tensión diferencial (CAN_H - CAN_L) es del orden de 2.0 V con CAN_H = 3.5V y CAN_L = 1.5V (nominales).

- Recesivo: la tensión diferencial (CAN_H - CAN_L) es del orden de 0V con CAN_H = CAN_L = 2.5V (nominales).

El estándar permite trabajar a una velocidad máxima de 1Mbit/segundo, en cuyo caso la longitud máxima del bus es de 40m, permitiendo conectar hasta 30 estaciones con una distancia mínima entre ellas de 0.3m, en la tabla No.1 se observa la relaciones de tiempo distancia y velocidad. Este estándar se encuentra implementado en Circuitos Integrados que se consiguen en el mercado del silicio, tales como el **PCA82C250** de **Philips** y el **UC5350** de **Unitrode,** propiedad de la **Texas Instruments**.

Tabla 1. Velocidad Distancia en CAN

Velocidad	Tiempo de Bit	Longitud Máxima
1 Mbps	1 μS	40 m
800 Kbps	1,25 μS	50 m
500 Kbps	2 μS	100 m
250 Kbps	4 μS	250 m
125 Kbps	8 μS	500 m
50 Kbps	20 μS	1000 m
20 Kbps	50 μS	2500 m
10 Kbps	100 μS	5000 m

7.4 FUNCIONES BÁSICAS DEL PROTOCOLO CAN

7.4.1 Censado de Portadora Acceso Múltiple con Detección de Colisiones

El protocolo de comunicaciones CAN es un protocolo CSMA/CD. La sigla CSMA **Carrier Sense Multiple Access,** significan que todo nodo en la red debe monitorear el bus en busca de un periodo de inactividad antes de tratar de enviar un mensaje al bus (**Carrier Sense**). También, una vez este periodo de inactividad ocurra, todo nodo en el bus tiene igual oportunidad para transmitir algún mensaje (**Multiple Access**). La sigla CD (**Collision Detection)** indica que si dos (o más) nodos en la red comienzan a transmitir al mismo tiempo, los nodos detectaran la colisión y harán la acción apropiada.

En el protocolo CAN se utiliza un método de **Arbitraje de bit no Destructivo**. Esto significa que el mensaje permanece intacto después de completar el arbitraje, incluso si se han detectado colisiones. Todo este arbitraje sucede sin corrupción o retardo del mensaje de mayor prioridad.

Hay un par de cosas que se requieren para soportar el **Arbitraje de bit no Destructivo**. Primero, estados lógicos necesarios para definir dominante o recesivo. Segundo, el nodo transmisor debe monitorear el estado del bus para ver si el estado lógico que trata de enviar actualmente aparece en el bus.

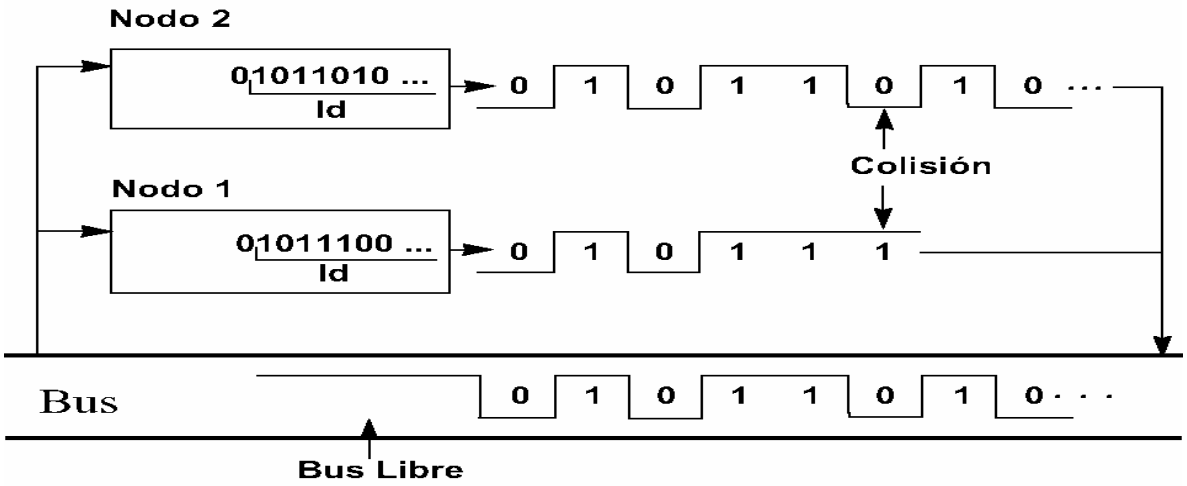
CAN define un bit **"0"** lógico como un bit dominante y un bit **"1"** lógico como un bit recesivo.

Un bit dominante siempre ganara el arbitraje sobre un bit recesivo, debido a esto el menor valor en el Identificador del Mensaje (es el campo usado en el proceso de arbitraje de mensaje), tiene mayor prioridad.

Como ejemplo, suponga dos nodos tratando de transmitir un mensaje al mismo tiempo. Cada nodo chequea el bus para asegurarse que el bit que esta tratando de enviar actualmente aparece en el bus. El mensaje de menor prioridad intentara en algún punto transmitir un bit recesivo entonces al chequear el bus el estado será dominante. En este punto este nodo pierde el arbitraje e inmediatamente detiene la transmisión. El mensaje de mayor prioridad continuara hasta completar la transmisión y el nodo que perdió el arbitraje esperara el siguiente periodo de inactividad en el bus e intentara transmitir su mensaje nuevamente (Ver Figura No. 2).

Este modo de resolver la contención es muy conveniente a la hora de desarrollar aplicaciones de tiempo real, pues permite contar con una plataforma cuyo comportamiento es predecible.

Figura 2. Arbitraje para la detección y resolución de colisiones.



7.4.2 Comunicación basada en mensajes

El protocolo CAN es un protocolo basado en mensajes, no es un protocolo basado en direcciones. Esto significa que los mensajes no son transmitidos de un nodo a otro basados en direcciones. Incluido en el mensaje CAN esta la prioridad y el contenido de los datos transmitidos. Todos los nodos en el sistema recibirán cada mensaje transmitido al bus (y reconocerán si el mensaje fue recibido correctamente). Depende de cada nodo en el sistema decidir si el mensaje recibido debe ser descartado o mantenido para ser procesado. Un solo mensaje puede ser destinado para que un nodo en particular lo reciba, o muchos nodos lo hagan, esto depende de la forma como la red y el sistema estén diseñados.
Esta característica es de gran importancia para el desarrollo de sistemas de control distribuido, pues asegura que la información entre todos los nodos del sistema es consistente.

Por ejemplo, un sensor de airbag de un automóvil puede ser conectado vía CAN solo hacia un nodo enrutador del sistema de seguridad. Este nodo enrutador toma información de otros sistemas de seguridad y enruta esta hacia otros nodos en la red del sistema de seguridad. Luego los otros nodos en la red del sistema de seguridad pueden recibir la información del sensor de airbag del enrutador al mismo tiempo que se verifica si el mensaje fue recibido apropiadamente y se decide si utiliza esta información o se descarta.

Otra característica útil construida dentro del protocolo CAN es la habilidad para que un nodo solicite información de otro nodo. Esto es llamado **Requerimiento de Transmisión Remota** (RTR). Esto es diferente al ejemplo del párrafo anterior, puesto que en lugar de esperar información enviada por un nodo en particular, el nodo específicamente solicita datos para que se envíen a este.

Por ejemplo, el sistema de seguridad de un carro toma frecuentemente actualizaciones de los sensores críticos como el de los airbags, pero este podría no recibir actualizaciones frecuentes de sensores como el sensor de presión de aceite o el sensor de batería baja, para

asegurarse de que estos funcionan apropiadamente, periódicamente, el sistema de seguridad puede solicitar datos desde estos otros sensores y hacer un chequeo a través del sistema de seguridad. El diseñador del sistema puede utilizar esta característica para minimizar el tráfico de la red mientras se mantiene la integridad de la red.

Un beneficio adicional de este protocolo basado en mensajes, es que se pueden adicionar otros nodos al sistema sin necesidad de reprogramar los otros nodos para reconocer esta adición. Este nuevo nodo empezara a recibir mensajes de la red y basado en los ID de los mensajes, decide si procesa o descarta la información recibida.

7.5 FILTRADO DE LOS MENSAJES CAN

7.5.1 Descripción de las tramas de los mensajes can

El protocolo CAN define cuatro diferentes tipos de mensajes (o Tramas). La primera y más común es la Trama de Datos (ver Figura No.3 y Figura No.4). Esta trama es usada cuando un nodo transmite información a cualquier otro nodo en el sistema. La segunda es la Trama Remota, la cual es básicamente una trama de datos con el bit RTR puesto en uno significando que es una Solicitud de Transmisión Remota (ver Figura No.4 para detalles de la Trama Remota). Los otros dos tipos de tramas son para manejo de errores. Una es llamada Trama de Error y la otra es llamada Trama de Sobrecarga (ver Figura No. 6 y Figura No. 7). Las Tramas de Error son generadas por los nodos que detecten cualquier error de los definidos por CAN. La Trama de Sobrecarga se genera por los nodos que requieren mas tiempo para procesar mensajes ya recibidos.

Figura 3. Trama de Datos Estándar.

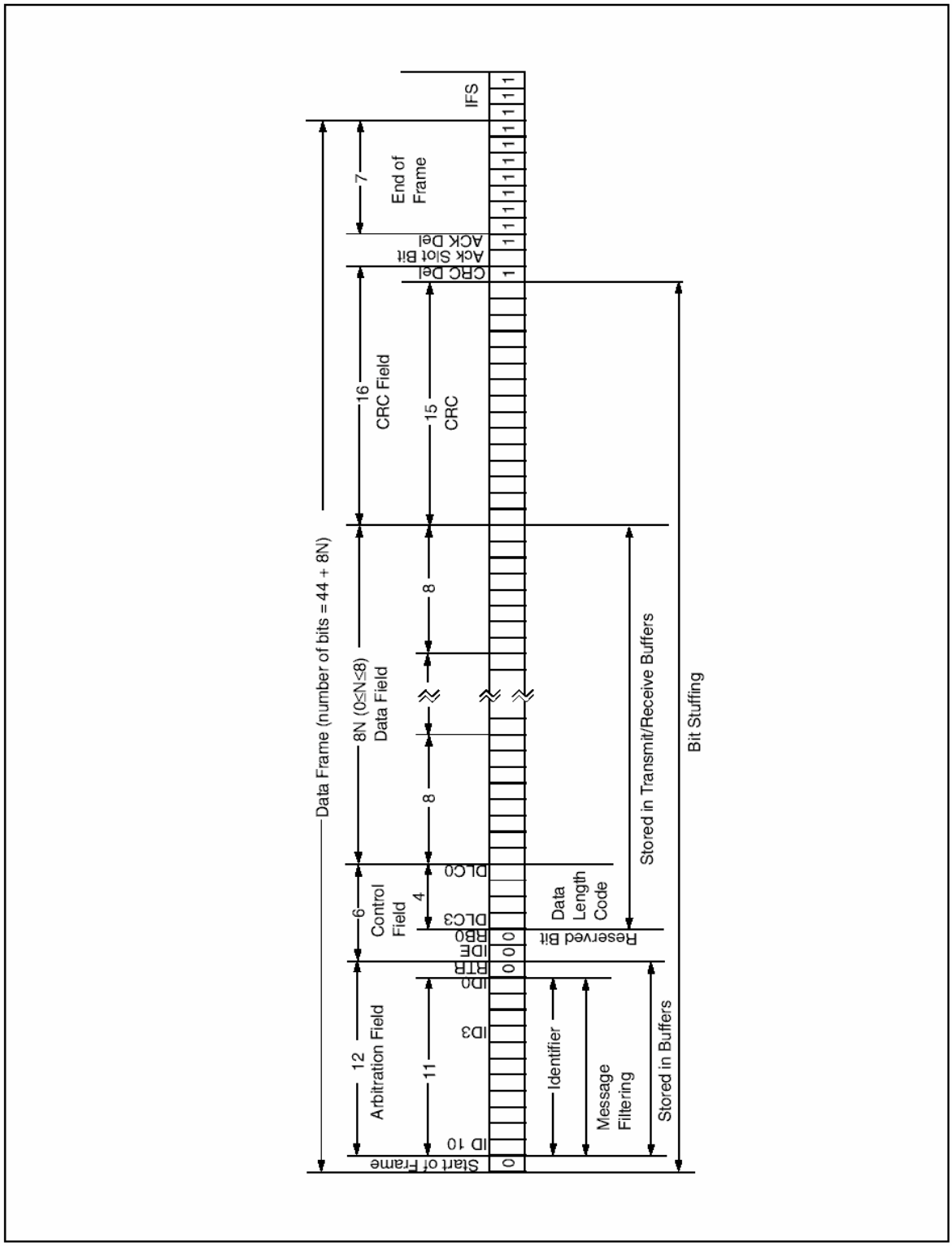


Figura 4. Trama de Datos Extendida.

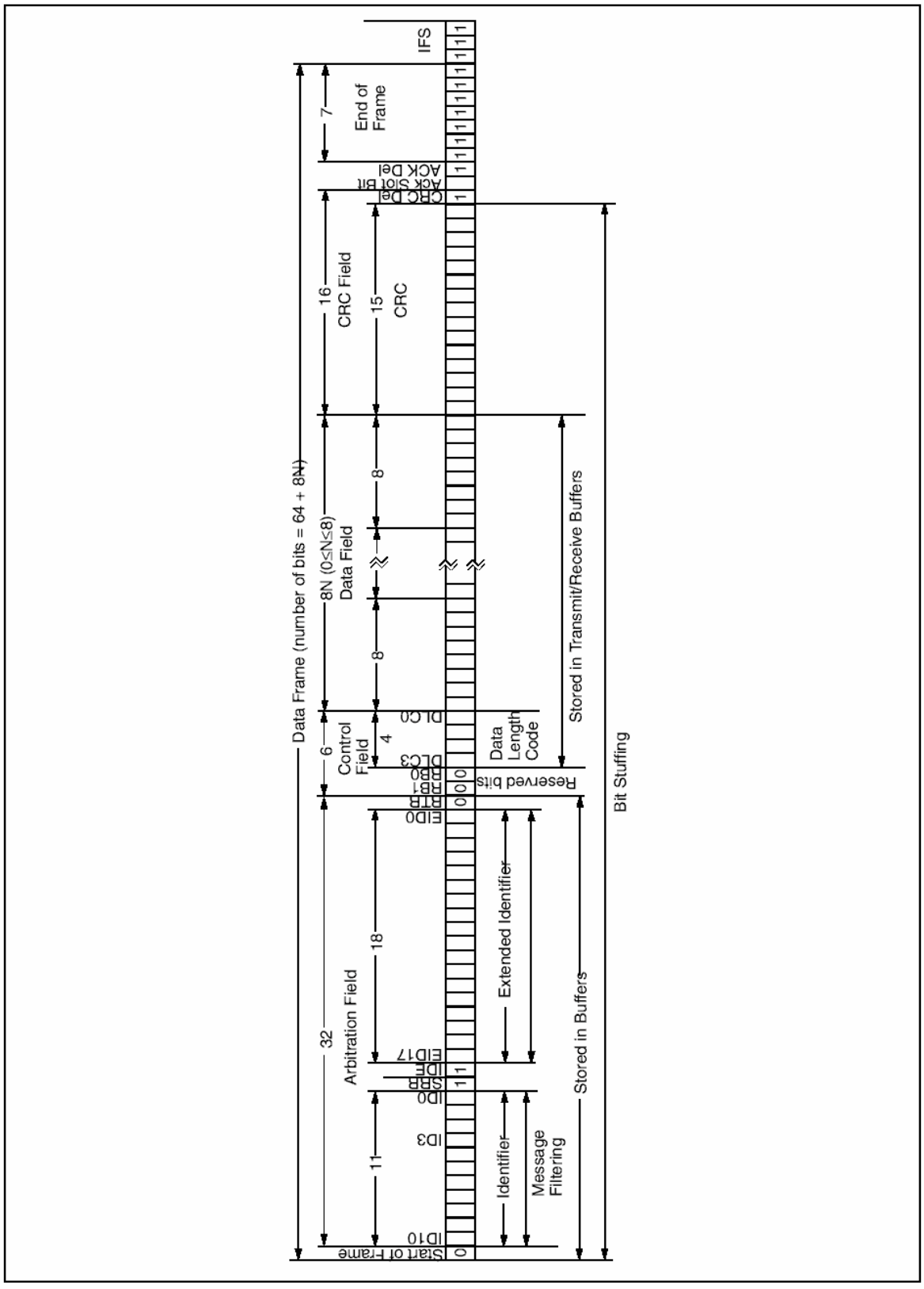


Figura 5. Trama de Requerimiento Remoto de Datos

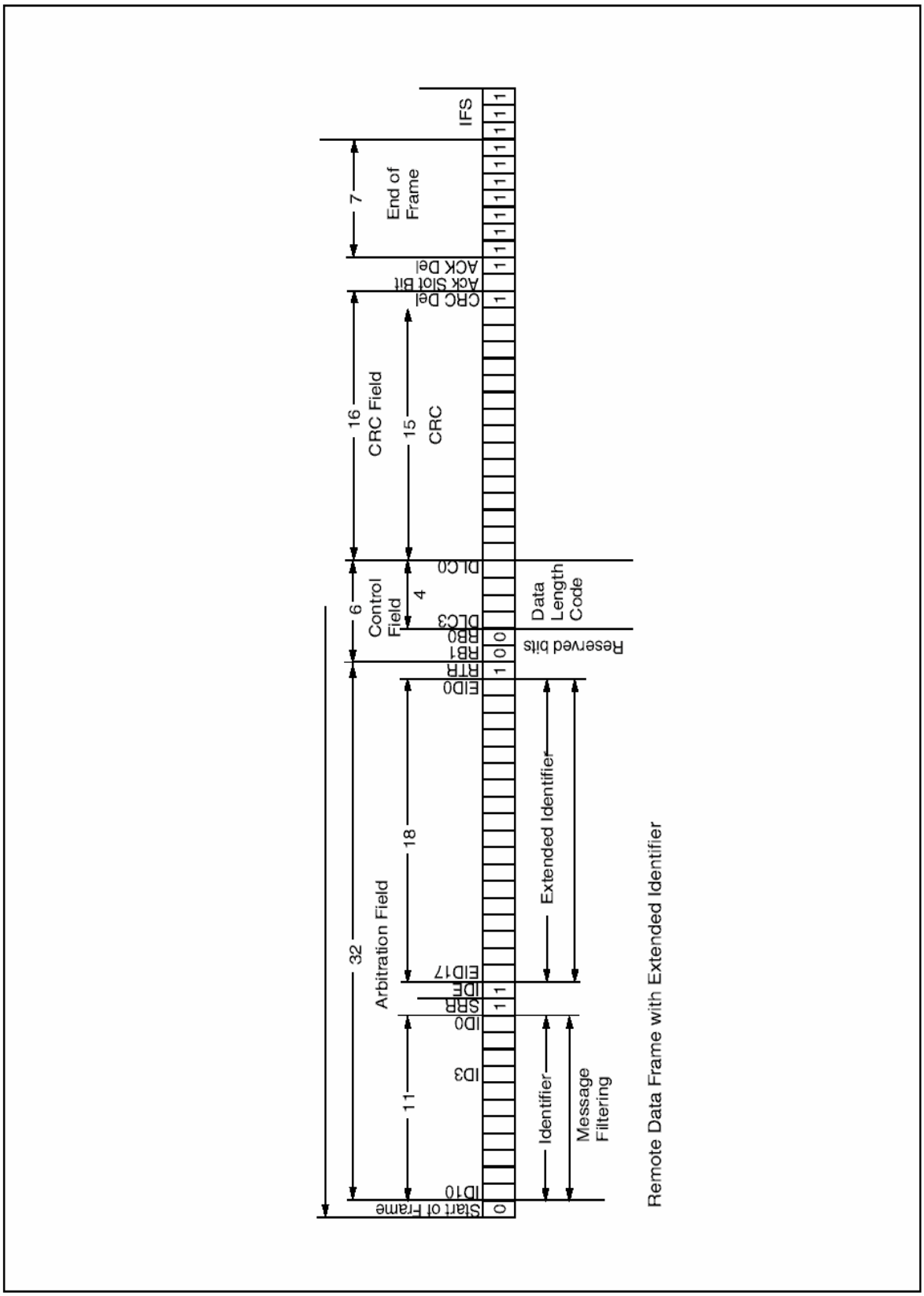


Figura 6. Trama de Error.

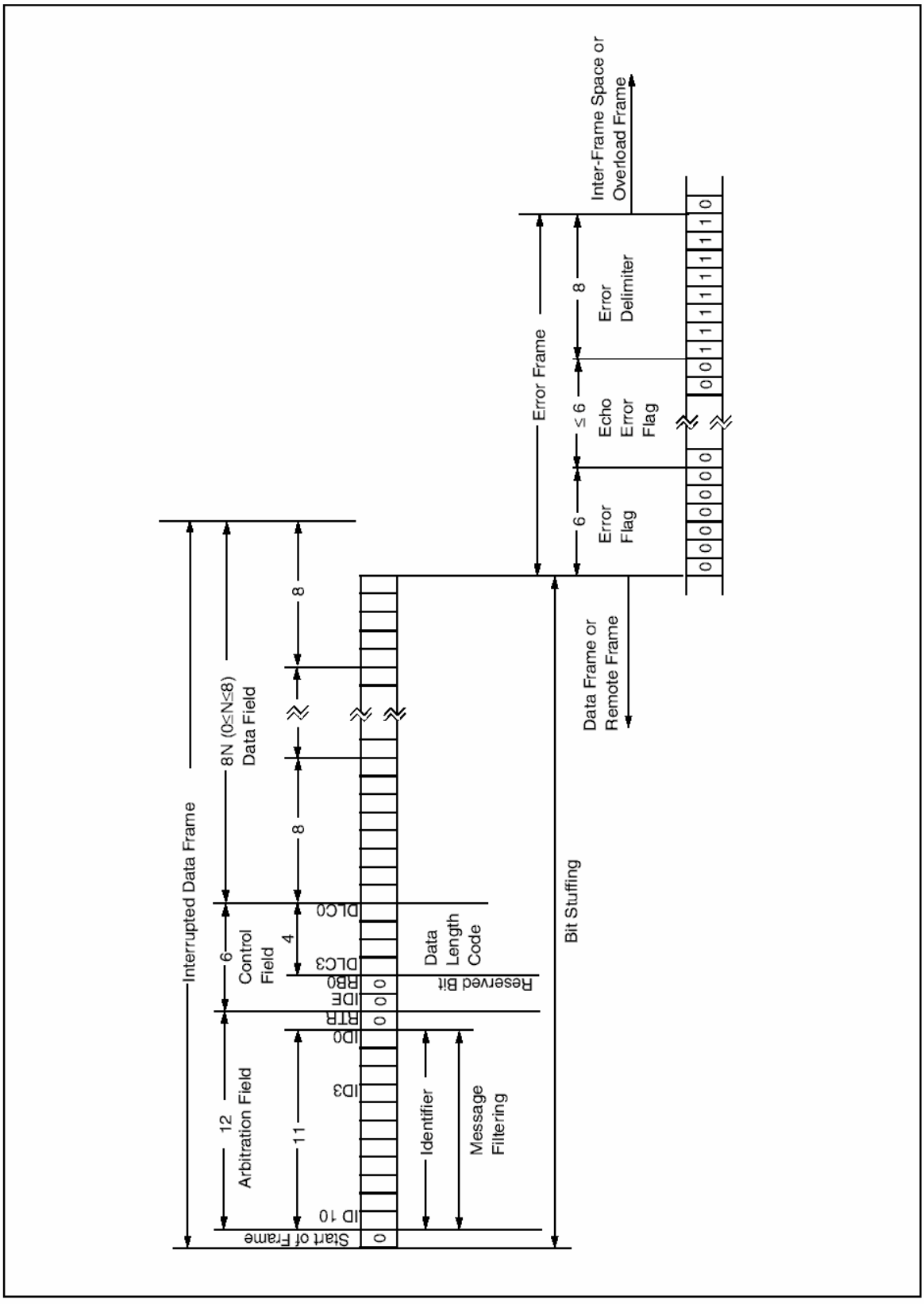
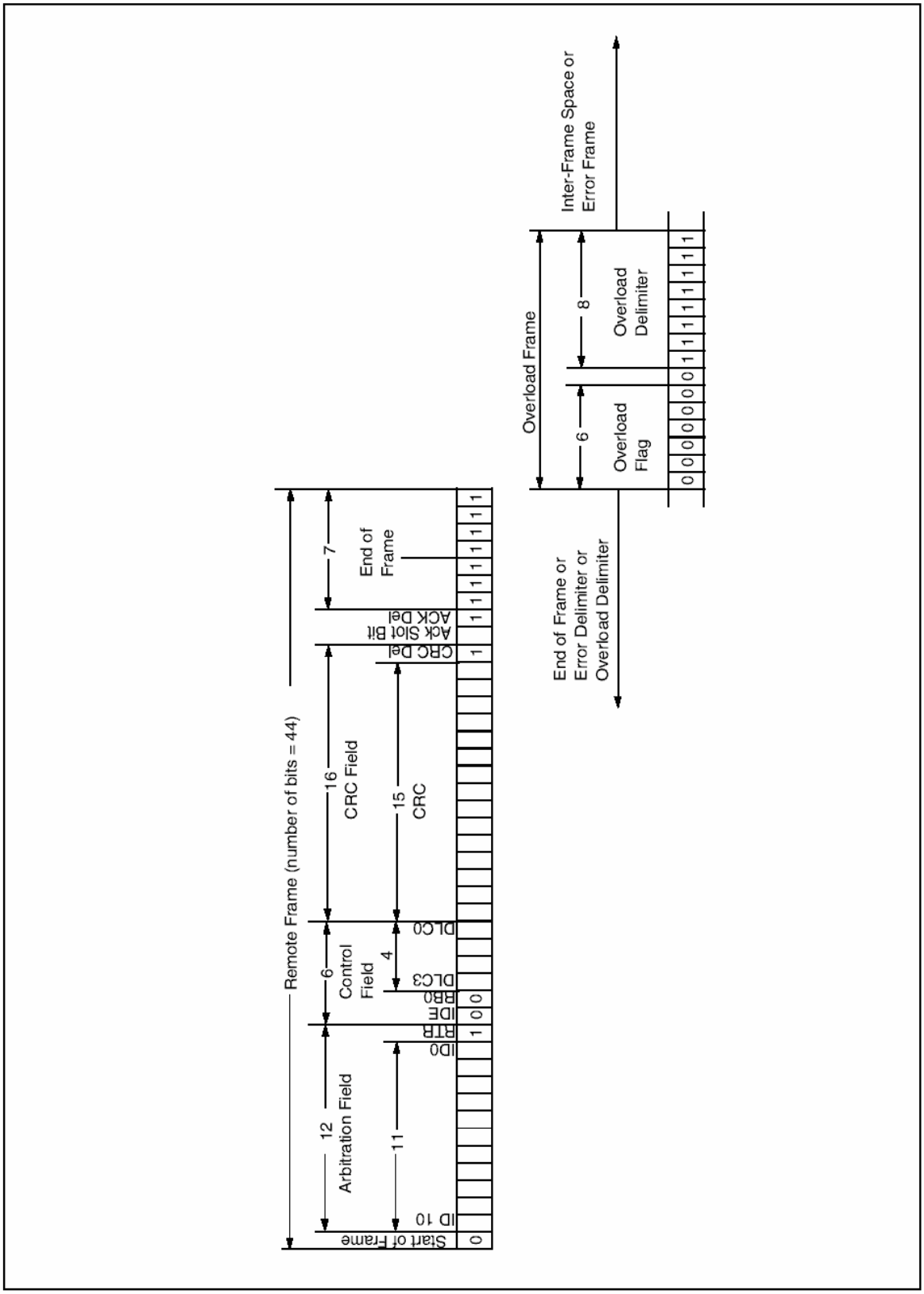


Figura 7. Trama de Sobrecarga.



La Trama de Datos consiste de campos que proveen información adicional sobre el mensaje como se define por las especificaciones CAN. Incluido en la Trama de Datos esta el Campo de Arbitraje, el Campo de Control, el Campo de Datos, el Campo CRC, un Campo de 2-bit de Reconocimiento y un Fin de Trama.

El campo de Arbitraje es usado para dar la prioridad a los mensajes en el bus. Desde que el protocolo CAN define un “0” lógico como un estado dominante, un número pequeño en el campo de arbitraje, tiene mayor prioridad en el bus. El campo de arbitraje cosiste de 12-bits (11 bits del identificador y un bit RTR) o 32 bits (29 bits del identificador, 1-bit para definir que el mensaje es una trama de datos extendida, un SRR bit el cual no es usado y un RTR bit), dependiendo en si utilizan Tramas Estándar o Tramas Extendidas. La versión actual de la especificación CAN, versión 2.0B, define identificadores de 29-bits y se llaman Tramas Extendidas. Versiones anteriores de las especificaciones CAN definen identificadores de 11-bits, las cuales son llamadas Tramas Estándar.

Como se describió en la sección anterior, la Solicitud de Transmisión Remota (RTR) es usada por un nodo cuando este requiere información enviada desde otro nodo. Para lograr un RTR, una Trama Remota es enviada con el identificador de la Trama de Datos requerida. El RTR bit en el Campo de Arbitraje es utilizado para diferenciar entre una Trama Remota y una Trama de Datos. Si el RTR bit es recesivo, entonces el mensaje es una Trama Remota. Si el RTR bit es dominante, el mensaje es una Trama de Datos.

El Campo de Control esta compuesto por seis bits. El MSB es el IDE bit (significa Trama Extendida) el cual debe ser dominante para Tramas de Datos Estándar. Este bit determina si el mensaje es una Trama Estándar o Extendida. En las Tramas Extendidas, el bit RB1 es un bit reservado. El siguiente bit es RB0 y también es reservado. Los cuatro LSBs son los bits DLC (Data Length Code – Código de Tamaño de Dato). Estos determinan cuantos bytes de datos están incluidos en el mensaje. Es de notar que una Trama Remota no tiene campo de datos, independiente del valor de los bits DLC.

El Campo de Datos consiste un número de bytes de datos descritos en el DLC del Campo de Control.

El Campo CRC consiste de 15-bits CRC y un delimitador CRC y es usado por los nodos receptores para determinar si ocurrieron errores de transmisión.

El Campo de Reconocimiento es utilizado para indicar si el mensaje fue recibido correctamente. Cualquier nodo que reciba correctamente el mensaje, sin importar si el nodo procesa o descarta los datos, pone un bit dominante en la ranura ACK durante un tiempo de bit (ver Figura No.3 o Figura No.4 para la ubicación de la ranura ACK).

Los dos últimos tipos de mensajes son Tramas de Error y Tramas de Sobrecarga. Cuando un nodo detecta alguno de los tipos de errores definidos por el protocolo CAN, una Trama de Error es generada. Una Trama de Sobrecarga dice a la red que el nodo que envía esta Trama, no esta listo para recibir un mensaje adicional en ese momento, o que la intermisión ha sido violada. Estos errores serán discutidos con más detalle en la próxima sección.

7.6 COMUNICACIÓN ROBUSTA Y RÁPIDA

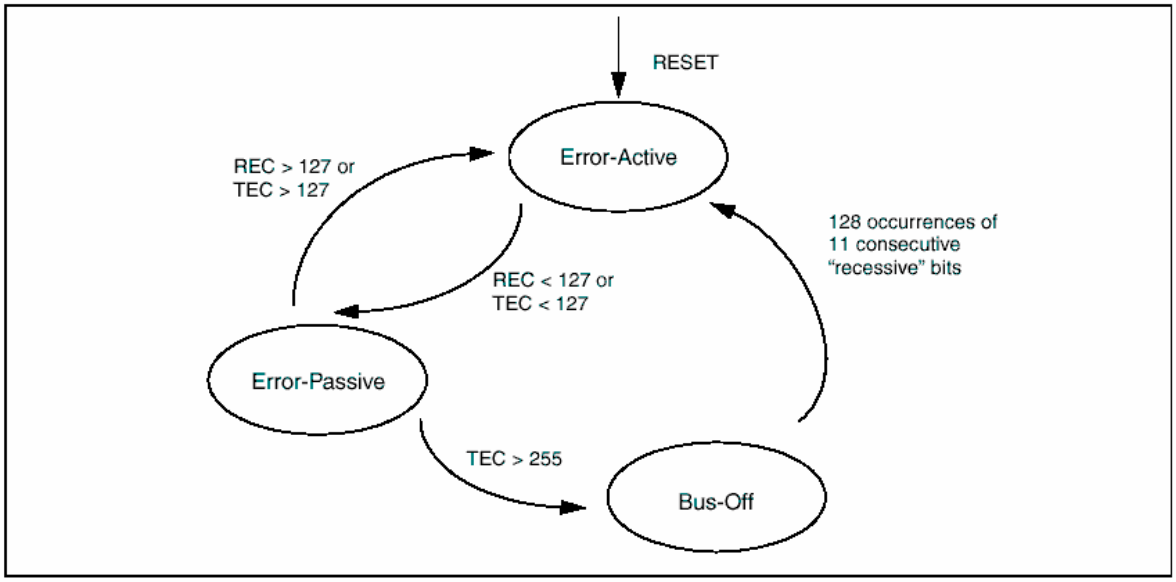
Debido a que CAN fue inicialmente diseñado para usarse en automóviles, un protocolo que maneja errores eficientemente fue crítico para que este ganara aceptación en el mercado. Con la salida de la versión 2.0B de las especificaciones CAN, la máxima velocidad de comunicación tuvo un incremento de 8 veces con respecto a la versión 1.0 hasta 1Mbit/seg. A esta velocidad, incluso los parámetros mas críticos en tiempo, pueden ser transmitidos en forma serial sin preocupare por retardos. Adicionalmente, el protocolo CAN tiene una lista comprensiva de errores que puede detectar y asegurar la integridad de los mensajes.

Los nodos CAN tienen la habilidad para determinar condiciones de fallo y transición de diferentes tipo, basados en la severidad de los problemas encontrados. También tienen la

habilidad para detectar pequeños disturbios o daños permanentes y modificar su funcionalidad adecuadamente.

Los nodos CAN pueden cambiar su funcionamiento de nodo normal (habilitado para transmitir y recibir mensajes normalmente), a apagado totalmente (bus-off) basado en la severidad de los errores detectados (Ver figura No.8). Esta característica es llamada confinamiento por falla. Ningún nodo CAN esta habilitado para monopolizar todo el ancho de banda de la red debido a fallos, en caso de que se presente esta falla, este nodo será confinado como un nodo fallido, el cual será apagado antes de que haga caer la red. Esto es muy importante debido a que el confinamiento por fallo garantiza el ancho de banda para información critica del sistema.

Figura 8. Diagrama de Estados para manejo de errores.



7.7 ESTADOS DE ERROR

Los errores son reconocidos por los nodos a través de las Tramas de Error o Bandera de Error. La transmisión de un mensaje erróneo es abortada y la trama es repetida tan pronto como el mensaje pueda otra vez ganar el arbitraje en la red. Un nodo con problemas se

puede encontrar en uno de tres posibles estados de error, Error-Activo, Error-Pasivo o Bus-Apagado.

7.7.1 Error-Activo

En Error-Activo el nodo puede tomar parte en las comunicaciones del bus activamente, incluyendo el envió de banderas de error activo, las cuales consisten de seis bits dominantes consecutivos. La Bandera de Error Activo viola la regla de inserción de bit y causa que los otros nodos envíen una Bandera de Error, llamada Bandera de Eco del Error, como respuesta. Una Bandera de Error-Activo y la Bandera de Eco del Error puede generar hasta doce bits dominantes consecutivos en el bus. Un nodo esta en Error Activo cuando el Contador de Errores de Transmisión (TEC) y el Contador de Errores de Recepción (REC) están por debajo de 128. Esto indica que el nodo esta en modo normal de operación, permitiendo al nodo transmitir y recibir sin restricciones.

7.7.2 Error-Pasivo

Un nodo pasa a Error-Pasivo cuando cualquiera, el Contador de Errores de Transmisión o el Contador de Errores de Recepción, exceden 127. Los nodos que se encuentren en Error-Pasivo no transmiten Banderas de Error-Activo en el bus, pero en cambio, transmiten Banderas de Error-Pasivo, las cuales consisten de seis bits recesivos consecutivos. Si el nodo en Error-Pasivo es el único transmisor en el bus, entonces la bandera de error pasivo violara la regla de inserción de bit y los nodos receptores responderán con sus propias Banderas de Error (activa o pasiva dependiendo de su estado de error).

Si el nodo en Error-Pasivo en cuestión no es el único transmisor (Ej. Durante el Arbitraje) o es un receptor, entonces la Bandera de Error Pasivo no tendrá efecto en el bus debido a su naturaleza recesiva. Cuando un nodo en Error-Pasivo transmite una Bandera de Error Pasiva y detecta un bit dominante, este solo vera el bus inactivo por ocho tiempos de bit adicionales después de una intermisión antes de reconocer el bus como disponible. Después de este tiempo, este intentara retransmitir.

7.7.3 Bus-Apagado

Un nodo entra en el estado de Bus-Apagado cuando el Contador de Errores de Transmisión es mayor que 255 (errores de recepción no pueden causar el estado de Bus-Apagado en el nodo). En este modo, el nodo no puede enviar o recibir mensajes, mensajes de reconocimiento o transmitir Tramas de Error de cualquier clase. Así es como se hace el Confinamiento por Fallo. Hay una secuencia de recuperación del bus definida por el protocolo CAN que permite recuperar a un nodo que esta en Bus-Apagado, retornando a Error-Activo, comenzado a transmitir de nuevo, si la condición de fallo es removida.

7.8 DETECCIÓN DE ERRORES

7.8.1 CRC Error

Cuando se envía un dato por medio de CAN, un valor de Chequeo de Redundancia Cíclica (CRC) de 15-bits es calculado por el nodo transmisor y este valor de 15-bit es transmitido en el campo CRC, todo nodo en la red recibe este mensaje, calcula el CRC del mensaje enviado y verifica que sea igual al recibido. Si no es así, un Error CRC ocurre y una Trama de Error es generada. Siempre que al menos un nodo no reciba el mensaje apropiadamente, el mensaje es reenviado después del tiempo de intermisión.

7.8.2 Error de Reconocimiento

En el Campo de Reconocimiento de un mensaje, los nodos transmisores chequean si la Ranura de Reconocimiento (el cual es enviado como un bit recesivo) contiene un bit dominante. Este bit dominante reconocerá que al menos un nodo recibió el mensaje correctamente. Si este bit es recesivo, entonces ningún nodo recibió el mensaje apropiadamente. Entonces se produce Un Error de Reconocimiento y se genera una Trama de Error y el mensaje original es retransmitido después del tiempo de intermisión.

7.8.3 Error de Forma

Si cualquier nodo detecta un bit dominante en uno de los siguientes cuatro segmentos del mensaje: Fin de Trama, Espacio entre Tramas, Delimitador de Reconocimiento o Delimitador CRC, el protocolo CAN define esto como violación de forma y un Error de Forma se genera. El mensaje original es entonces reenviado después del tiempo de intermisión. (Ver Figura No.3 y/o Figura No.4 para ubicar estos segmentos en el mensaje CAN).

7.8.4 Error de Bit

Un Error de Bit ocurre si un transmisor envía un bit dominante y detecta un bit recesivo, o si es enviado un bit recesivo y se detecta un bit dominante cuando se chequea el nivel actual en el bus y se compara este con el bit que fue enviado. En el caso donde el transmisor envía un bit recesivo y se detecta un bit dominante durante el Campo de Arbitraje o la Ranura de Reconocimiento, no se genera Error de Bit debido a que ocurre un arbitraje normal o un reconocimiento. Si un Error de Bit es detectado, una Trama de Error es generada y el mensaje original es reenviado después del tiempo de intermisión.

7.8.5 Error de Inserción

El protocolo CAN usa el método de transmisión No Retorno a Cero (NRZ). Esto significa que el nivel del bit es colocado en el bus por un tiempo de bit completo. CAN es también asincrónico y la inserción de bit es usada para permitir a los nodos receptores sincronizarse recuperando la información de reloj del flujo de datos. Los nodos receptores se sincronizan en la transición de recesivo a dominante. Si hay más de cinco bits de la misma polaridad en fila, CAN automáticamente inserta un bit con polaridad opuesta en el flujo de datos. Los nodos receptores usan este para la sincronización, pero ignora el bit insertado en los datos. Si, entre el Inicio de Trama y el Delimitador CRC, se detectan seis bits consecutivos con la misma polaridad, entonces la regla de Inserción de Bit ha sido violada. Entonces ocurre un Error de Inserción, se envía una Trama de Error y el mensaje es repetido.

7.9 CONCLUSIÓN

El protocolo CAN fue optimizado para sistemas que necesitan transmitir y recibir pequeñas cantidades de información (comparado con Ethernet o USB, los cuales están diseñados para mover grandes bloques de datos) confiable a cualquier otro nodo en la red. CSMA/CD permite a todos los nodos igual oportunidad para ganar acceso al bus y un manejo limpio de las colisiones.

Partiendo de que el protocolo CAN este basado en mensajes, no en direcciones, todos los nodos en el bus reciben todos los mensajes y reconocen estos, independientemente de si necesita o no los datos contenidos en el mensaje. Esto permite al bus operar con mensajes en formato nodo a nodo o multicastin sin tener que enviar diferente tipos de mensajes.

CAN fue concebido como un protocolo de alta seguridad. Para ello se han adoptado medidas adecuadas en cada una de las capas de protocolo: En la capa física la disponibilidad de transceptores con capacidad de funcionamiento en condiciones degradadas. Todos los mensajes transmitidos son reconocidos de forma consistente por los receptores enviando una trama con bit ACK que se transmite como recesivo. En las tramas de datos e interrogación remota se aplica la regla de relleno de bits que evita una secuencia sucesiva de más de 5 bits del mismo signo, para ello se inserta un sexto bit *de signo* contrario, el receptor ha de eliminar este bit adicional siguiendo la misma regla. Para detección de errores se incluye un código CRC con distancia Hamming 6, la tasa de error no detectado es menor que (tasa de error en mensajes)*$4{,}7 \times 10^{-11}$. Cualquier nodo que detecta un error transmite una trama que señala el error a los demás nodos, si el nodo detector es un nodo totalmente activo (no se encuentra en nivel pasivo de error) el mensaje queda invalidado para toda la red y se retransmitirá lo antes posible. El tiempo de recuperación es como máximo de 29 veces el tiempo de bit. Se sigue un sofisticado proceso de diagnóstico en los nodos, cuando un nodo acumula errores pasa inicialmente a una situación de funcionamiento pasivo y si la degradación continúa el nodo queda excluido de la comunicación evitando perturbar al resto de nodos de la red. Es decir el estado de un

nodo puede ser: Activo, Pasivo o Anulado. Un nodo anulado ha de deshabilitar su transceptor y no participa en la comunicación. CAN a alcanzado un nivel extraordinario de madurez e implantación, se habla se cientos de millones de nodos, los fabricantes y procesadores digitales de señal están incorporando controladores CAN de forma bastante generalizada. Lo modelos VHDL de controladores CAN se pueden incorporar en ASICs y dispositivos de lógica programable (FPGAs). CAN resulta una opción a tener en cuenta en sistemas distribuidos de tiempo real.

Las razones por la cual se eligió CAN para el desarrollo de este proyecto son las siguientes:

- Su alta seguridad en la trasmisión de datos en ambientes ruidosos (industria).
- La consistencia de la información lo hace un protocolo versátil para aplicaciones en tiempo real.
- Su acogida por parte de los fabricantes de chips que ya tienen el protocolo CAN introducido en sus microcontroladores o en chips individuales, como lo es en nuestro caso el MCP2510 de la microchip.
- La siguiente tabla nos da una comparación de las características físicas de Bus CAN con otros protocolos de BUS que usan RS485.

Tabla 2. Comparación de CAN con protocolos que usan el estándar RS485.

BUSES	Velocidad	No. Nodos	Tamaño de Datos	Distancia máxima
ASI Bitbus Profibus DP estándar DIN E 19245 T3	9,6 Kbps a 12MBits	32 Participantes por Segmento	4 Bits 43 Bytes 246 Bytes 1000 Bits	1200 m a 100 m
BUS CAN	10Kbps a 1Mbps	32 0 64 Participantes por segmento	64 Bytes	5000 m a 40 m

8 HARDWARE NECESARIO PARA EL PROYECTO

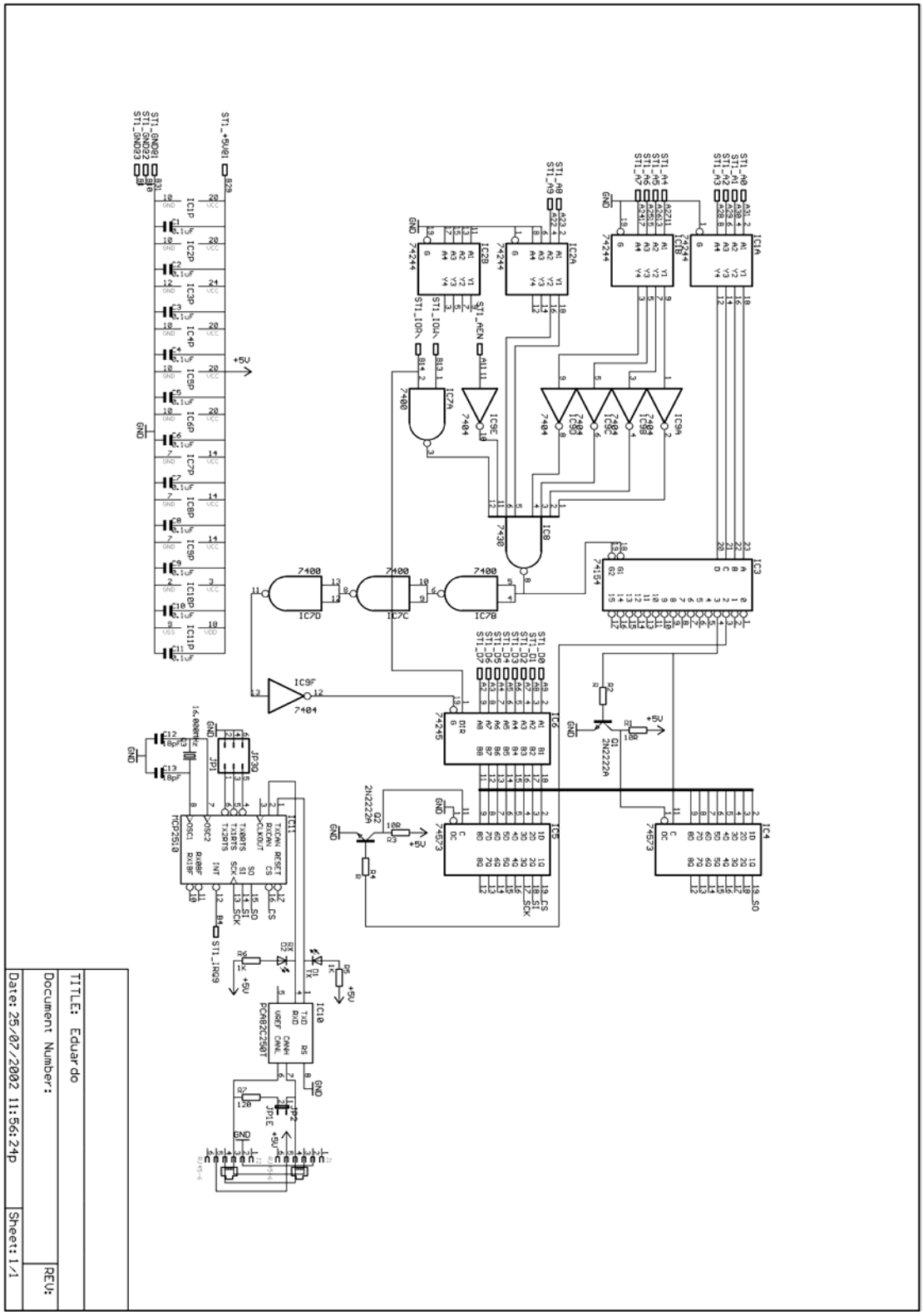
8.1 INTERFACE ISA PARA LA COMUNICACIÓN ENTRE EL BUS INDUSTRIAL Y LA COMPUTADORA

Este parte se encarga de enlazar la comunicación de la computadora con Red CAN, para esto se utiliza una de las ranuras de expansión tipo ISA de la computadora. Como se puede apreciar en la figura No. 9 este sistema esta conformado por un conjunto de integrados que cumplen con funciones de protección en el caso de los buffer 74244 y de direccionamiento en el caso del decodificador 74154 en conjunto con los las compuertas NAND y los negadores, quienes se encargan de habilitar el integrado 74255, que es un buffer bidireccional que permite hacer la lectura y escritura de los datos sobre los Laches 74573, que a su vez son controlados por el decodificador.

El direccionamiento es hecho a partir de la compuerta 7430 y los últimos 4 bits del bus de direcciones ISA, pues la compuerta habilita el decodificador 74154, para que la tarjeta reciba o envíe datos al bus ISA cuando los bits A9 hasta A4 sean 110000 y el bit AEN (encargado de habilitar el direccionamiento externo en la interface ISA) este activo, es decir, “0”, ya que este es activo en bajo. Los anterior nos permite conocer las direcciones de acceso a los dispositivos que halla en la tarjeta pues los primeros 4 bits (A3 hasta A0) actúan en las entradas ABCD del decodificador, pudiendo variar de 0h hasta Fh. Uniendo los bits de direccionamiento ISA tenemos un rango de direcciones que va desde 300h hasta 30fh, con lo que se puede actuar con 16 dispositivos de entrada o salida conectados en la tarjeta ISA.

En el caso particular de comunicación por inteface SPI solo se usan dos pines del decodificador que equivalen a las direcciones 302 y 303, con los que se habilitan los laches 74573, el primero que introduce la señal S0 al bus de datos ISA y el segundo que saca las señales CS, SI y SCK desde el bus de datos ISA.

Figura 9. Esquema de la tarjeta ISA con el hardware CAN.



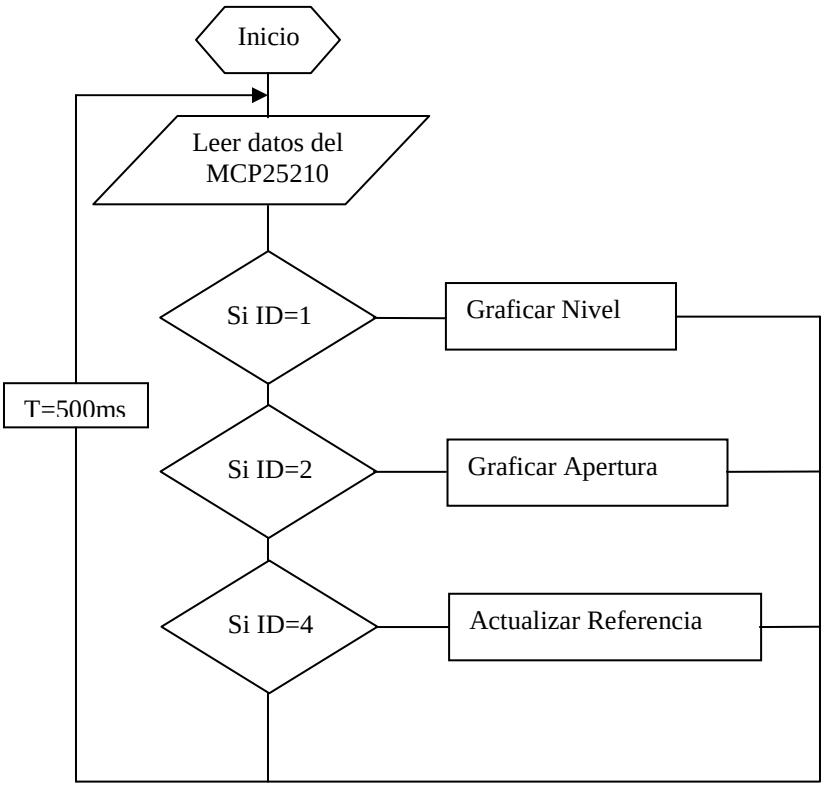
Lo anterior con el fin de hacer que los datos puedan ser transmitidos, a través de laches, que actúan como un par de puertos de ocho bits uno como entrada y otro como salida. Usando este mecanismo se implementó un protocolo de comunicación serial SPI, que se encargara de comunicar el chip MCP2510 con la computadora.

Para la implementación del protocolo SPI se cuenta con una Librería de Datos Dinámica o DLL que se ha desarrollado en Visual C++, y usando el lenguaje de programación ensamblador para los procesadores INTEL.

La librería DLL, llamada SPI.dll, es utilizada por el software de Monitoreo y Manipulación del sistema para configurar los diferentes registros del chip CAN MCP2510 de la Microchip. Teniendo en cuenta esta información podremos asegurar la comunicación entre el Sistema de Control de Nivel y la computadora.

Como se vio en el capitulo 7, las Tramas CAN deben tener un identificador que sirve para especificar la prioridad y el contenido de la información encapsulada en la trama. Estos identificadores son utilizados por el programa en Visual Basic para saber que dato esta llegando desde la red. Para el caso del sistema de control de nivel, tenemos 3 variables que son: El Nivel del Tanque, El Porcentaje de Apertura de la Válvula y El Nivel Deseado o Nivel de Referencia. Estas variables son etiquetadas con 3 identificadores distintos para poder distinguir estas en el momento de leer los datos desde el chip MCP2510, con el cual estamos conectados en la Red CAN, de la cual se reciben las variables desde el medidor de ultrasonido y la Válvula Inteligentes. De estos dos dispositivos se escribirá mas adelante. En el siguiente diagrama de flujo se especifica el procedimiento llevado a cabo.

Figura 10. Diagrama de flujo del programa en Visual Basic.



8.2 SISTEMA DE DESARROLLO PARA EL MICROCONTROLADOR A USAR EN LOS DISPOSITIVOS DE MEDICIÓN DE NIVEL Y CONTROL DE LA VÁLVULA.

En el pasado, las tecnologías existentes en el mercado obligaban a pequeños o medianos industriales a invertir mucho dinero en equipos adicionales al microcontrolador, como son los borradores ultravioleta, programadores, y en algunos casos, en un buen software que les permitiera editar, ensamblar, simular, realizar conexiones y programarlo. Los nuevos microcontroladores y las herramientas de desarrollo disponibles para los usuarios de 8 bits de **MOTOROLA**, permiten todas estas facilidades mediante un programa interno de fábrica, que permite iniciar proyectos de forma muy sencilla y rápida y sin ningún costo adicional.

El MC68HC908JK3 es uno de los miembros de la familia 08 de Motorola, caracterizados por su bajo costo y alto desempeño. Todos los miembros de esta familia utilizan la unidad de procesamiento denominada CPU08 y están disponibles en una gran variedad de presentaciones (20, 28 y 40 pines) y en diversos tamaños de memoria de programa (1.5K, 4K y 32K). Dentro de las principales características de estos microcontroladores es que son de memoria Flash (borrable y programable eléctricamente) además de contar con 10 canales de conversión de analógico a digital (ADC). En la tabla 1 se pueden observar las características más importantes de la CPU08 y en la tabla 2 las características del microcontrolador MC68HC908JK3.

Tabla 3. Características de la CPU08 de Motorola

Características de la CPU08
Un modelo de programación muy completo, con registro de 16 bits.
Set de instrucciones muy amplio y varios modos de direccionamiento.
Registro de 16 bits y stack pointer manipulable por el usuario.
Instrucciones de transferencia de Memoria a Memoria
Instrucciones de Multiplicación rápida de 8x8
Instrucciones de División rápida de 16/8
Instrucciones de BCD (Binary Coded Decimal)
Fácil soporte de lenguajes de alto nivel como el C.

Tabla 4. Características del Microcontrolador MC68HC908JK3 de Motorola

Características del MCU MC68HC908JK3
CPU 08 de 8 bits
Operación interna a 8 MHz
Rango de operación desde 3V
LVI: protección de bajo voltaje.
4 K Bytes para memoria de programa (FLASH)
128 Bytes de memoria RAM
10 canales de análogo a digital de 8 bits
15 Entradas/Salidas
2 temporizadores (timers) de 16 bits (Salida por comparación y/o entrada de captura)
Software 100% compatible con el de la familia 05
Versiones HC (cristal o resonador) y HRC (Resistor y capacitor)
Modos de bajo consumo (STOP y WAIT)
COP (Watchdog): perro guardian
8 fuentes de Interrupciones (totalmente vectorizadas)
Pulldowns programables por puerto de entrada

8.3 ARQUITECTURA INTERNA DEL JK3 Y MODELO DE PROGRAMACIÓN DE LA CPU08

En la figura 11 se puede apreciar un diagrama interno del microcontrolador y en la figura 12 se muestran los registros internos de la CPU08 mediante los cuales se procesan las instrucciones, los operadores y los resultados. La función básica de cada uno de ellos se describe a continuación.

Acumulador (A): Es un registro de 8 bits de propósito general usado en las operaciones aritméticas y lógicas.

Registro índice (H:X): Es un registro de 16 bits utilizado como apuntador en el modo de direccionamiento indexado.

Puntero de pila (SP): Es un registro de 16 bits que contiene la dirección de la posición disponible en el *stack*.

Contador de programa (PC): Es un registro de 16 bits que contiene la dirección de la siguiente instrucción u operando a ser procesado.

Registro de banderas (CCR): Es un registro de 8 bits que contiene el bit de enmascarado general de interrupciones y 5 banderas de estado, las cuales indican ciertas condiciones originadas por la instrucción previamente ejecutada.

Figura 11. Diagrama de bloques internos del microcontrolador Motorola 68HC908JK3

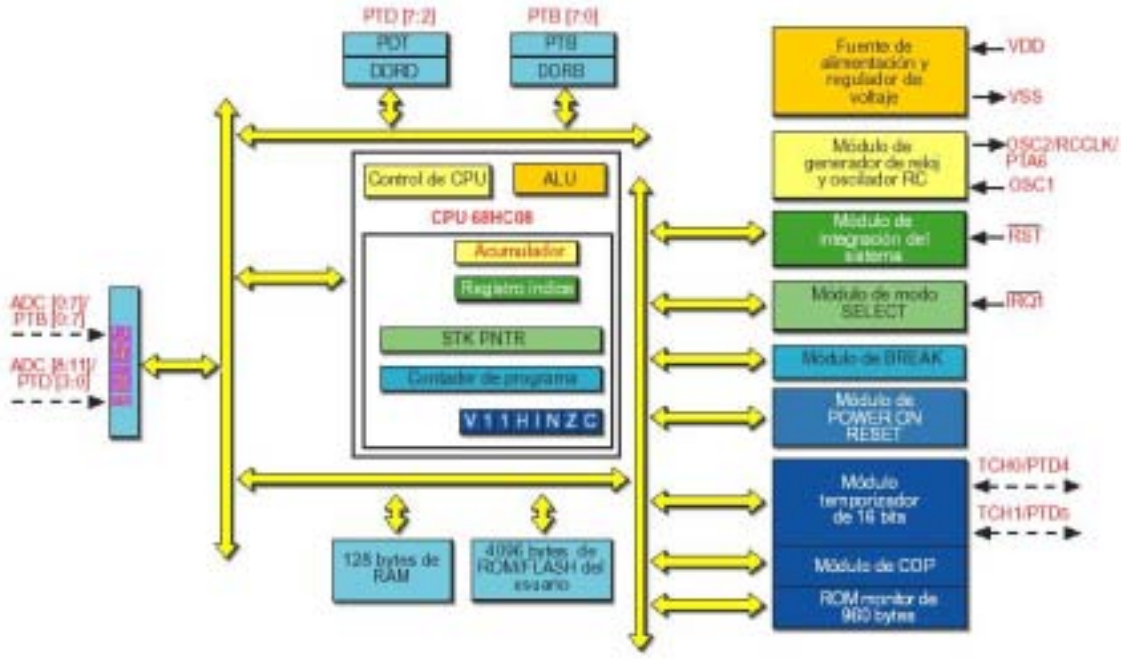
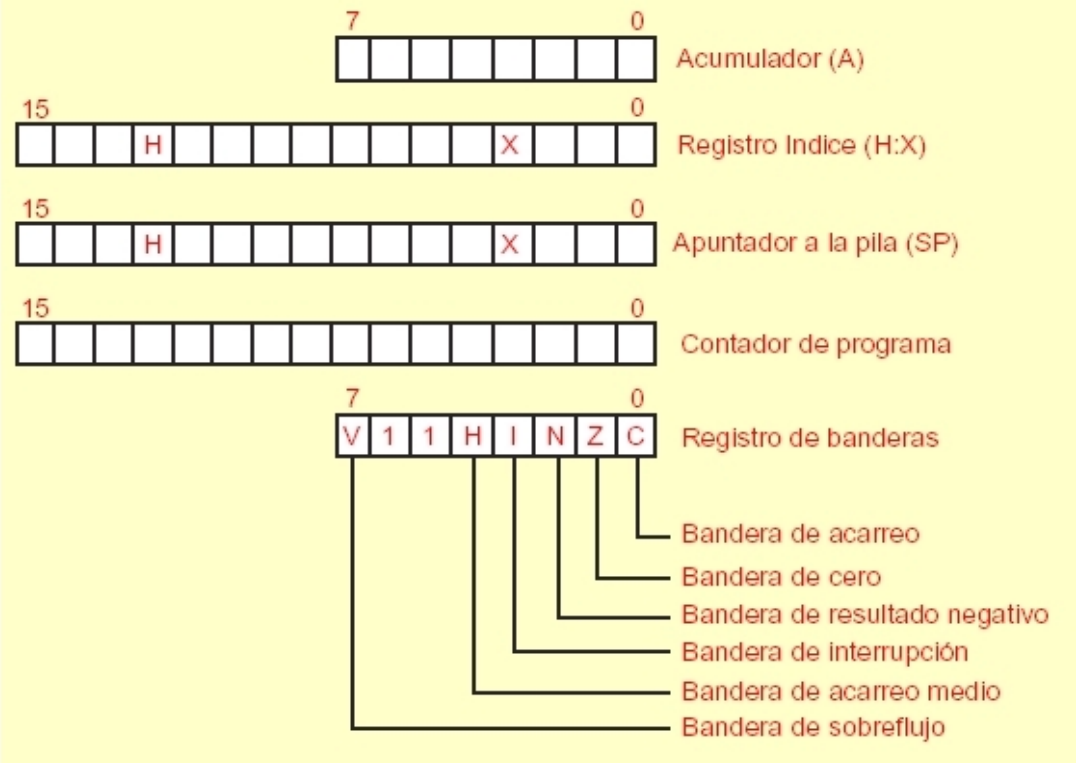


Figura 12. Estructura de los registros internos y ubicación de las banderas



8.3.1 Mapa de memoria

Aquí se encuentran las direcciones de los registros internos, la posición que ocupa la RAM y la parte asignada al programa del usuario. De igual manera, se observa la zona en la cual se encuentra almacenado el programa de fábrica (ROM MONITOR) que nos permite realizar simulación, *debug* y programación del microcontrolador. En la zona alta de memoria se encuentran los vectores de interrupción y en ellos se encuentra la dirección (2 bytes) con la cual se carga el contador de programa (PC) cuando se presenta alguna de las interrupciones.

8.3.2 Sistema de desarrollo

El sistema soporta los microcontroladores **JK3** y el **MC68HC908JK1** (**JK1**) de la familia **08** de **MOTOROLA**. En la figura 13 se observa la conexión básica con la computadora. El sistema consta de un software y de una tarjeta en la cual se inserta el microcontrolador con los elementos básicos para su funcionamiento (Ver figura 14). Posee entrada para alimentación de DC, conversores de nivel TTL a RS-232 que permiten conexión serial a una computadora para simulación y programación, circuito oscilador basado en cristal y LED rojo indicador de alimentación. El jumper J1 selecciona el modo de funcionamiento. En la posición «PROG» el sistema arranca en modo SIMULACION- PROGRAMACION, permitiendo mediante el software instalado (ICS08JLZ Development Kit de P&E Microcomputer Systems Inc.), realizar simulación y posterior programación del chip. J1 en la posición «APP» permite que el sistema arranque en modo aplicación; es decir, ejecuta el programa grabado por el usuario. SW1 es un suiche de dos polos y dos posiciones, que permite remover la alimentación completa del sistema cuando el software del PC así lo solicite para validar el código de seguridad interno del micro. El pulsador RST está conectado a la señal de reset del micro permitiendo el control por el usuario de esa función. En la figura 15, se puede observar la ubicación de los elementos sobre la tarjeta.

Figura 13. Conexión básica de la tarjeta con la computadora y con la aplicación

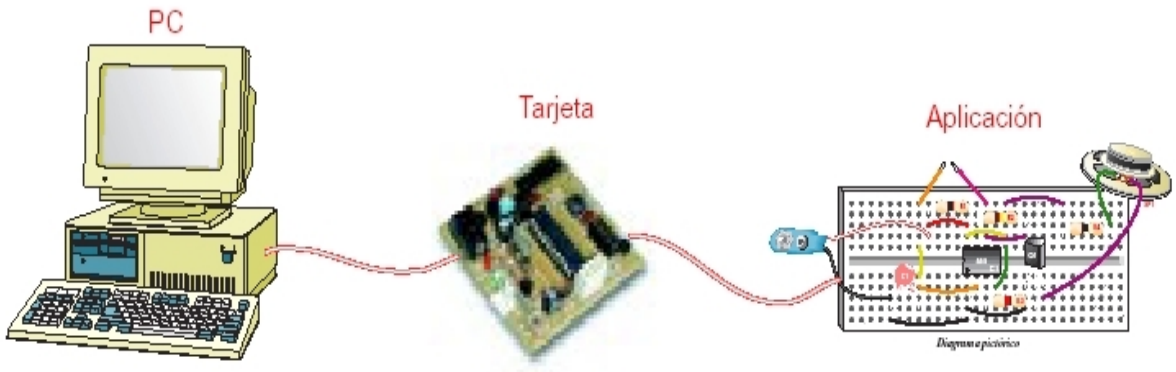


Figura 14. Circuito del Sistema de Desarrollo utilizado en el proyecto.

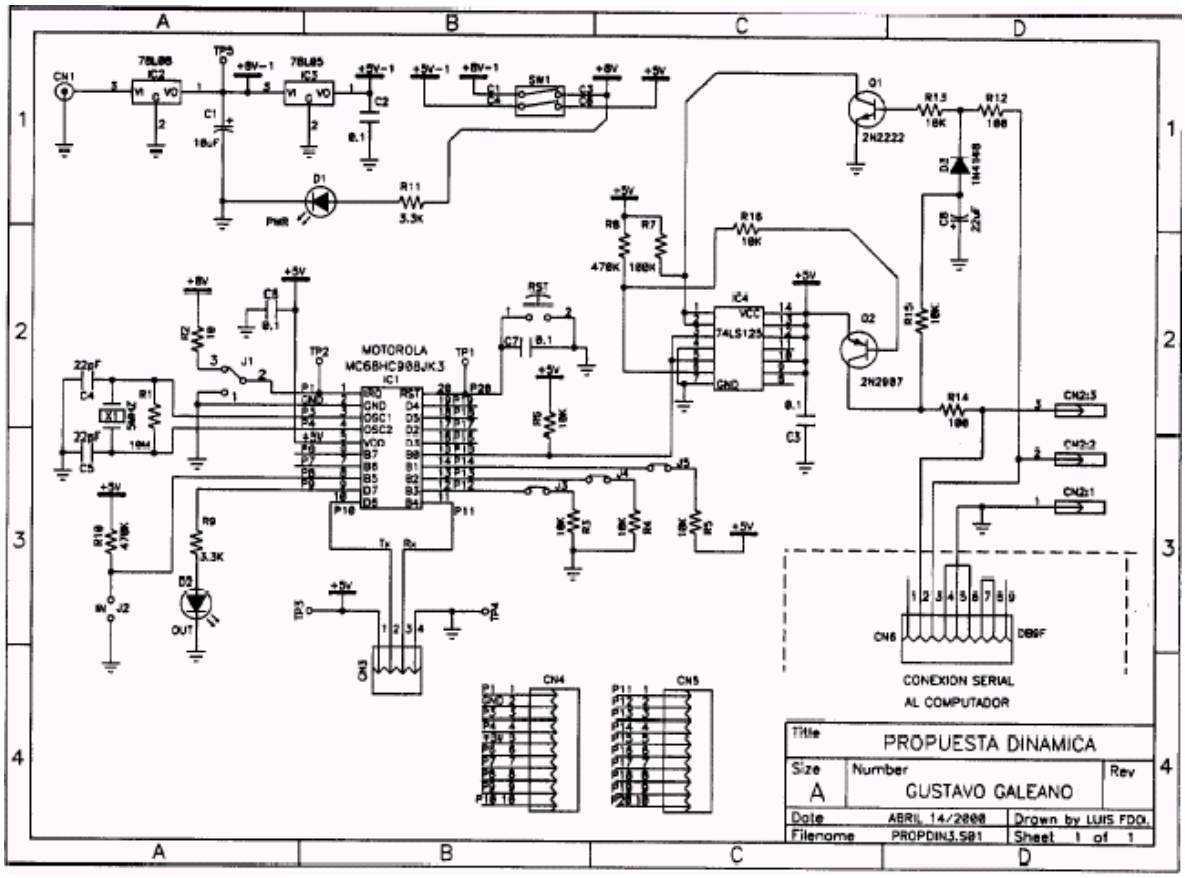
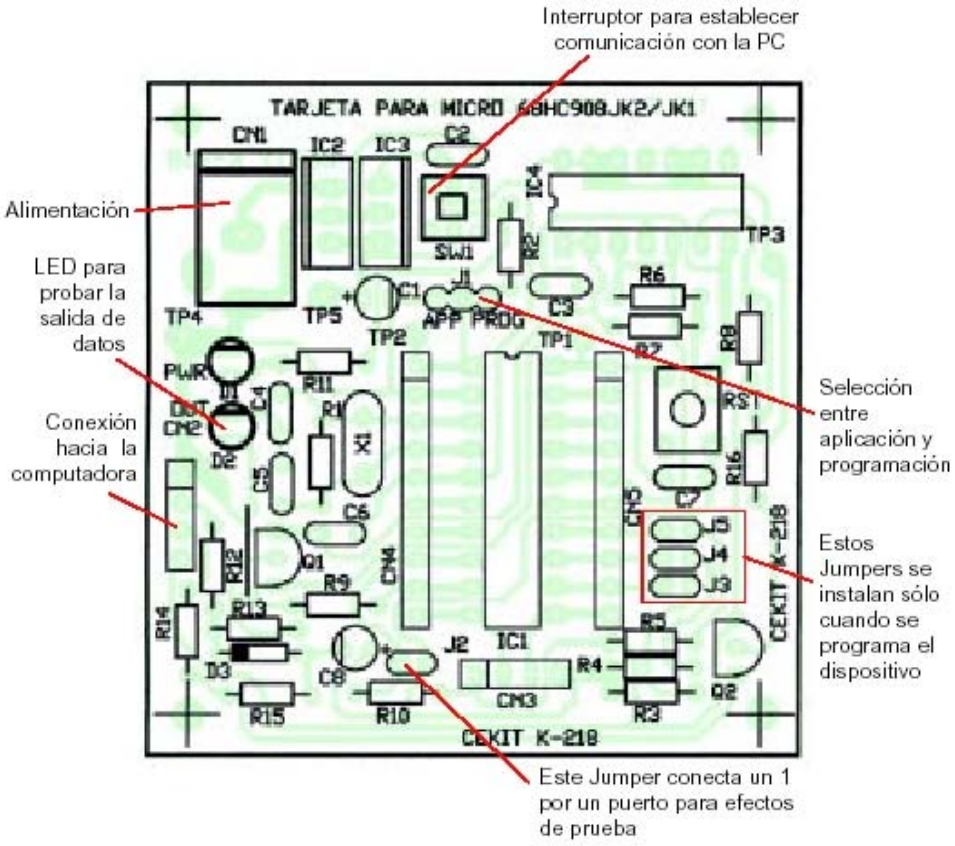


Figura 15. Montaje de los elementos sobre la tarjeta.

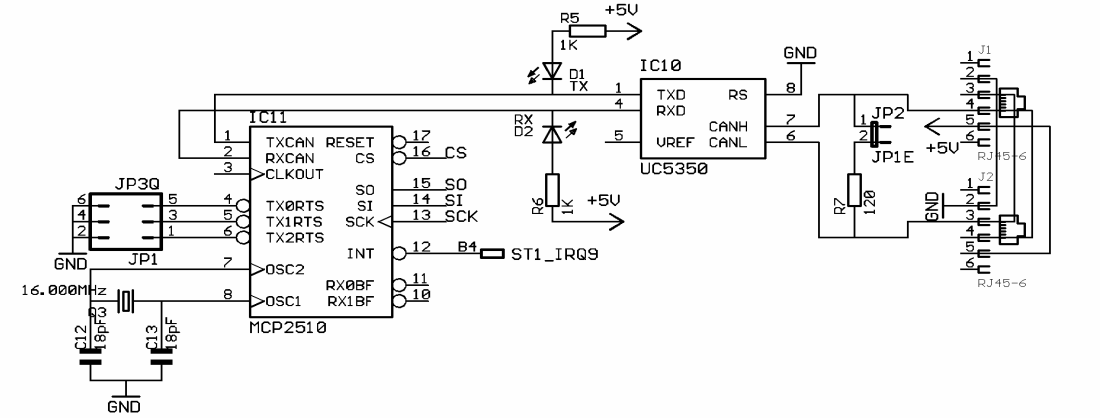


Para mayores detalles sobre el microcontrolador MC68HC908JK3, se puede consultar la documentación anexa a este proyecto.

8.4 MONTAJE DE LOS CHIPS DE COMUNICACIÓN PARA LA RED CAN.

Se utilizaron los chips MCP2510 de Microchip para implementar la capa de enlace de datos del protocolo CAN y los chips UC5350 Unitrode de Texas Instruments, para el nivel Físico de la Red CAN, para mayor información de cada uno de ellos se recomienda revisar las hojas de datos de ambos que se encuentran anexas en este Proyecto. En la figura No. 16 Se puede observar el esquema para el montaje de estos integrados.

Figura 16. Esquemático del montaje de los chip MCP2510 y UC5350.



La programación del chip CAN nivel de enlace de datos, se realizo a través del microcontrolador Motorola MC68HC908JK3 del que ya se ha hablado. A través de este chip se programaron los registros del MCP2510, diseñando unas rutinas en lenguaje ensamblador que implementan una interface SPI para la comunicación con MCP2510. La información modificada en los registros permitirá la y la intercambio de datos entre los módulos.

En la interface usuario se utilizo un programa en ensamblador para procesadores Intel 8086 para la modificación de los registros del MCP2510. Ver sección 7.1.

En la configuración de tres CNF1, CNF2, CNF3 de los registros del MCP2510, que son los encargados de determinar la velocidad de transferencias de Bus CAN se hizo uso del MCP2510 Bit Timing Calculator, software desarrollado por la Intrepid Control Systems, Inc. (este se puede encontrar en los anexos del Proyecto). Los resultados son generados en forma de reporte en HTML como se puede ver a continuación.

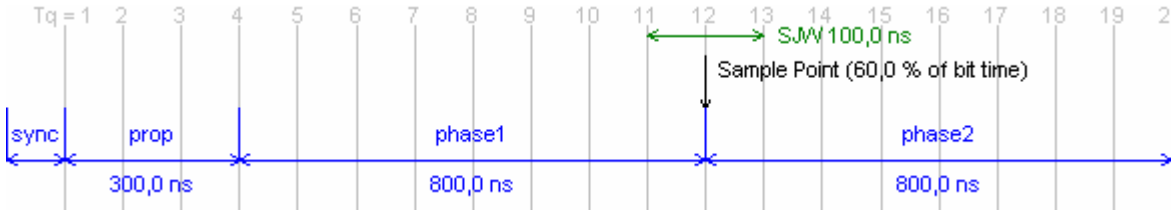
Setup Criteria	
Oscilator Frequency	20,000 MHz
Target CAN Bus Baud Rate	500,000 kbps

Selected Options	
BRP-1 (Baud Rate Prescaler)	0
Tq (Time Quanta)	100,000 ns
Number of Time Quanta	20
% Error of Target Baud Rate	0,0 %

Bit Timing Setup in Tq	
Propagation Delay	3
Phase Segment 1	8
Phase Segment 2	8
Syncronization Jump Width (SJW)	1

Multiple bit sampling is off. Wakeup filter is off.

Bit Timing Diagram



Configuration Register Setup		
Register	Binary	Hexadecimal
CNF1	b'00000000'	0x00
CNF2	b'10111010'	0xBA
CNF3	b'00000111'	0x07

Generated 9:18:30 am 12/17/2003
MCP2510 Bit Timing Calculator by Intrepid Control Systems, Inc. (www.intrepidcs.com)

Para lograr una mejor compresión del chip MCP2510, se elaboró un programa en Visual Basic, que se encarga de chequear la información que esta en los registros del MCP2510

(ver tabla No.3) y hacer la manipulación de algunos registros que permiten la transmisión y recepción de datos extrayendo o colocando información en el MCP2510, El programa y el código fuente esta en los archivos anexos a esta proyecto.

El programa consta de funciones de configuración que permite hacer las pruebas necesarias para probar el funcionamiento del chip, pues el mismo chip contiene un registro que permite configurar los modos de operación entre los cuales esta el modo de retorno que permite recibir la información que se envía para probar el estado de los chip tanto del MCP2510 como del UC5350. También se cuenta con un registro de banderas que permite verificar los diferentes errores que se puedan presentar en el Bus CAN y determinar si han llegado datos al chip. Generando también interrupciones para diferentes fuentes o entradas que pueden ser chequeadas en un registro dedicado a esta función. Para mayor información del funcionamiento de cada registro es recomendable ver la hoja de datos del MCP2510 que se incluye en los anexos de este proyecto.

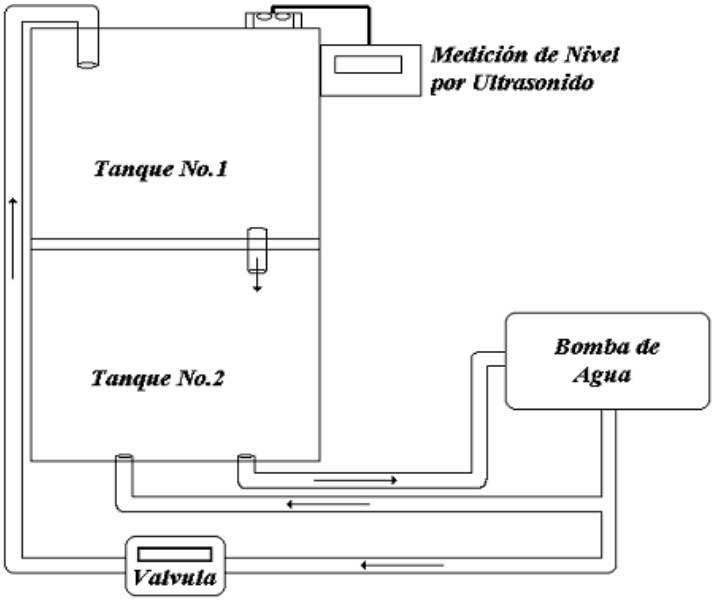
Tabla 5. Mapa de registros del MCP2510.

Lower Address Bits	Higher Order Address Bits							
	x000 xxxx	x001 xxxx	x010 xxxx	x0011 xxxx	x100 xxxx	x101 xxxx	x110 xxxx	x111 xxxx
0000	RXF0SIDH	RXF3SIDH	RXM0SIDH	TXB0CTRL	TXB1CTRL	TXB2CTRL	RXB0CTRL	RXB1CTRL
0001	RXF0SIDL	RXF3SIDL	RXM0SIDL	TXB0SIDH	TXB1SIDH	TXB2SIDH	RXB0SIDH	RXB1SIDH
0010	RXF0EID8	RXF3EID8	RXM0EID8	TXB0SIDL	TXB1SIDL	TXB2SIDL	RXB0SIDL	RXB1SIDL
0011	RXF0EID0	RXF3EID0	RXM0EID0	TXB0EID8	TXB1EID8	TXB2EID8	RXB0EID8	RXB1EID8
0100	RXF1SIDH	RXF4SIDH	RXM1SIDH	TXB0EID0	TXB1EID0	TXB2EID0	RXB0EID0	RXB1EID0
0101	RXF1SIDL	RXF4SIDL	RXM1SIDL	TXB0DLC	TXB1DLC	TXB2DLC	RXB0DLC	RXB1DLC
0110	RXF1EID8	RXF4EID8	RXM1EID8	TXB0D0	TXB1D0	TXB2D0	RXB0D0	RXB1D0
0111	RXF1EID0	RXF4EID0	RXM1EID0	TXB0D1	TXB1D1	TXB2D1	RXB0D1	RXB1D1
1000	RXF2SIDH	RXF5SIDH	CNF3	TXB0D2	TXB1D2	TXB2D2	RXB0D2	RXB1D2
1001	RXF2SIDL	RXF5SIDL	CNF2	TXB0D3	TXB1D3	TXB2D3	RXB0D3	RXB1D3
1010	RXF2EID8	RXF5EID8	CNF1	TXB0D4	TXB1D4	TXB2D4	RXB0D4	RXB1D4
1011	RXF2EID0	RXF5EID0	CANINTE	TXB0D5	TXB1D5	TXB2D5	RXB0D5	RXB1D5
1100	BFPCTRL	TEC	CANINTF	TXB0D6	TXB1D6	TXB2D6	RXB0D6	RXB1D6
1101	TXRTSCTRL	REC	EFLG	TXB0D7	TXB1D7	TXB2D7	RXB0D7	RXB1D7
1110	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT
1111	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL

8.5 ACONDICIONAMIENTO DEL TANQUE DE NIVEL

Se preciso de una planta de nivel como la que muestra la figura No. 17, la cual tiene dos tanques de 25 cm de alto cada uno. En el tanque No.1 se lleva a cabo el control del nivel utilizando para ello los dos sistemas implementados (Medición de Nivel por Ultrasonido y Válvula Inteligente). El tanque No.2 servirá para almacenar el líquido que interviene en el proceso, esto es con el fin de simular una planta de control de nivel real.

Figura 17. Diagrama del Sistema de Control de Nivel.



8.5.1 Diseño del sistema de Medición de Nivel por Ultrasonido

Este sistema es el encargado de hacer la medición de nivel en el tanque No.1 (Figura No.17), posteriormente enviar esta información a la Red CAN. Esto se puede explicar a través del diagrama de bloques de la figura No.18 en la que se puede observar que se cuenta con dos elementos que se encargan de transmitir y de recibir las señales en el rango del ultrasonido que esta en la banda de los 40Khz. Cada receptor y transmisor posee un circuito de acondicionamiento, que es controlado por dos puertos del MC68HC908JK3, quien se encarga de generar un tren 10 pulsos a una frecuencia de 40Khz, en cuanto estos pulsos

regresan al rebotar sobre el liquido que hay en el tanque, el microcontrolador se encarga de recibirlos y determinar el tiempo que le tomo a los pulsos hacer el recorrido de ida y regreso.

Figura 18. Diagrama del Sistema medición de Nivel.

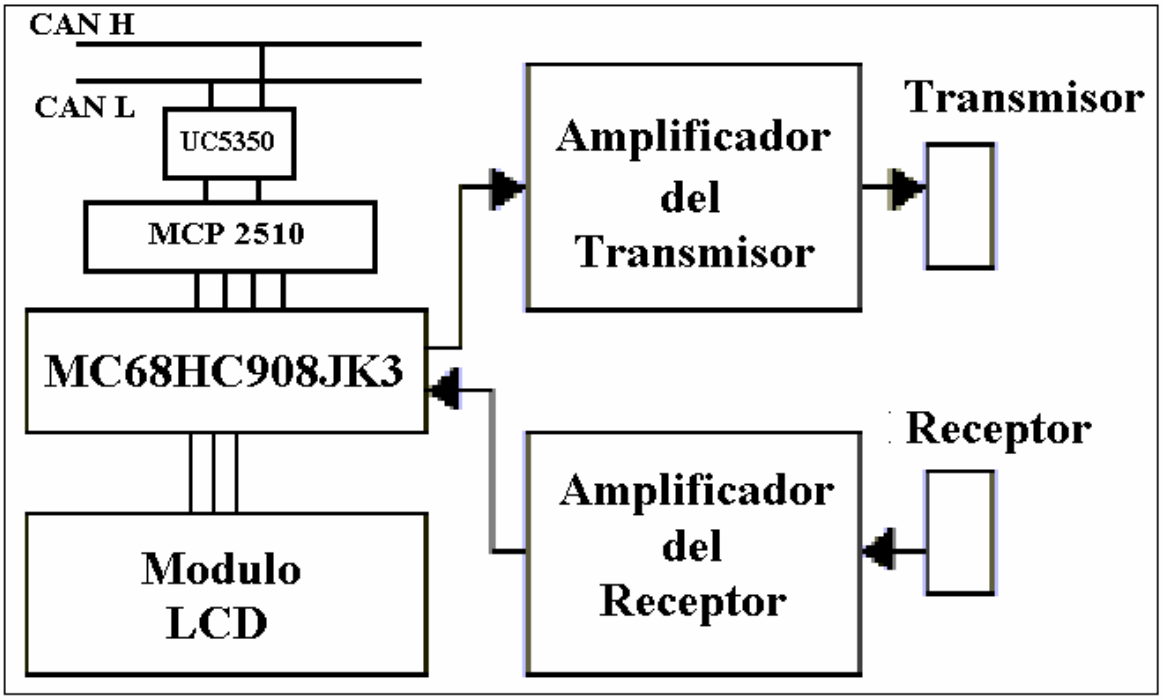
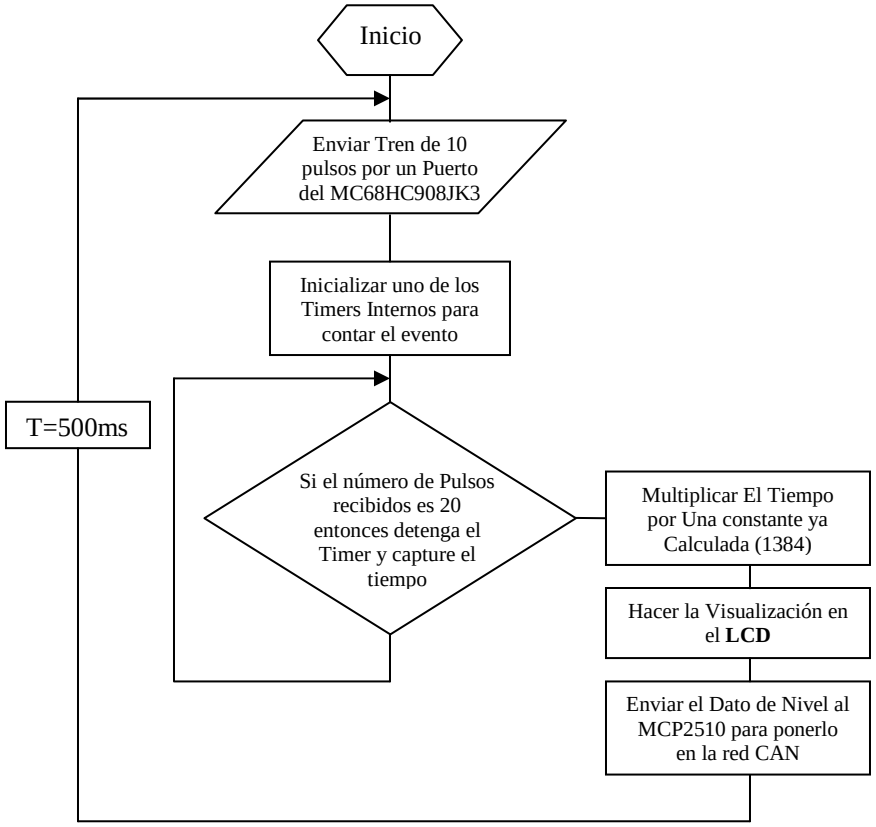


DIAGRAMA DE BLOQUES DEL MEDIDOR POR ULTRASONIDO

El tiempo que tarda es luego dividido por dos, para luego ser multiplicado con la velocidad del sonido que es aproximadamente de 340m/s. Entonces podemos obtener la distancia que hay entre el nivel del agua y el conjunto de sensores de ultrasonido, para que al final se haga la resta de la altura del tanque menos la distancia obtenida, con lo que tenemos la medida de nivel en el tanque No.1. El procedimiento anterior es llevado a cabo en el microcontrolador MC68HC908JK3, en el cual se han implementado las rutinas para operaciones con números de 32 bits, para que de esta manera se pueda obtener un valor del nivel muy aproximado.

El sistema esta dotado de una pantalla de LCD, que es la encargada de mostrar los datos adquiridos y calculados por el microcontrolador. Así que puede observarse siempre el nivel de líquido que hay en el tanque.

Figura 19. Diagrama de Flujo del programa del modulo de Ultrasonido



El tiempo que dura el sonido en ir y volver desde que parte del emisor hasta que es captado por el micrófono receptor, relacionado con la velocidad del sonido, proporciona la información necesaria para determinar la distancia entre el emisor y la superficie en la que se reflejó o rebotó la señal enviada. El diagrama de bloques del circuito se muestra en la figura No. 18 en donde se ven las etapas principales del proyecto. Este proyecto consiste básicamente de cuatros etapas funcionales:

- Emisora del tren de pulsos.
- Receptora.

- Procesamiento y cálculo.
- Visualización y transmisión.

Etapa emisora del tren de pulsos: Esta integrada por el microcontrolador ya descrito antes, el amplificador y el trasmisor ultrasónico 40TR16F (CASE:ALUMINUM/FLOWER) figura No. 21. El microcontrolador inicia la cuenta del tiempo y genera un tren de pulsos a una frecuencia de 40 KHz en el PTD5 del micro (Figura No. 14), esta es amplificada y se lleva al transductor ultrasónico el cual convierte el tren de pulsos eléctrico en ultrasonido.

Etapa receptora: Esta conformada por un receptor ultrasónico 40TR16P (CASE:PLASTIC/BLACK), un amplificador, un transistor figura No. 21 y el MC68HC908JK3 (Figura No. 18). El receptor convierte el tren de pulsos ultrasónico reflejados, en un tren de pulsos eléctricos, pasando al amplificador operacional y luego al transistor el cual entrega un voltaje compatible con la entrada del microcontrolador, cuando este detecta un cambio en el PTD4 del micro (Figura No. 14), genera una interrupción deteniendo el conteo de tiempo que se inició cuando el Micro generó el tren de pulsos.

Para las etapas emisora y receptora se tienen las ecuaciones de la figura No. 20 que permiten determinar las los valores de *Rf* y *Ri* para los dos amplificadores de la figura No. 21. Para el receptor se requiere de una ganancia de 450, esta nos dará una buena amplificación para la señal que se recibe. Para el transmisor con una ganancia de 45 será suficiente para excitar el sensor ultrasónico dando una buena potencia a nuestra aplicación. (Esto fue probado en el laboratorio con el osciloscopio).

Figura 20. Ecuaciones para un amplificador inversor.

$$Vo = -AVi \qquad A = \frac{Vo}{Vi}$$

$$Vo = -\frac{Rf}{Ri}Vi \qquad A = \frac{Rf}{Ri}$$

Se escoge una resistencia de 220Ω como resistencia de entrada a cada amplificador, para luego calcular la resistencia de realimentación usando las ecuaciones de la figura No.20.

Rfr = Ar*Rir ; Rfr = 450*220 = 99000 Ω ≈ 100KΩ

Rft=At*Rit ; Rfr = 45*220 = 9900 Ω ≈ 10KΩ

Etapa de procesamiento y calculo: El microcontrolador toma el valor del tmr0, lo pasa por una rutina de multiplicación de 32 bits luego es restado al valor de la altura del tanque, lo que entrega un valor de 0 a 8912760 μm (en Teoría ya que la distancia máxima que se puede medir depende del alcance del los sensores y la etapa de acondicionamiento) quedando solo por visualizar el resultado en metros para nuestro tanque se obtendrán valores entre 0-25 cm.

$v = \frac{d}{t}$ $d = v * t$ *v*: Velocidad del sonido: $346\,{}^{m}\!/_{S}$

Sabiendo que se hace un doble recorrido, es decir, la ida y el regreso de los pulsos. Se hace entonces una división por dos.

$$d_{recorrida} = \frac{346\,{}^{m}\!/_{S} * t}{2}$$ *Ecuación No. 1*

El microcontrolador tiene un cantador de eventos con un reloj de 1,25Mhz, lo cual en tiempo equivale a 0.8μS. El tiempo entonces esta dado por esta constante y el valor que captura el timer en el registro de 16 bits TCHT.

$$t = TCNT * 0.8\mu S$$

Reemplazando esta relación en la ecuación No. 1, tenemos que:

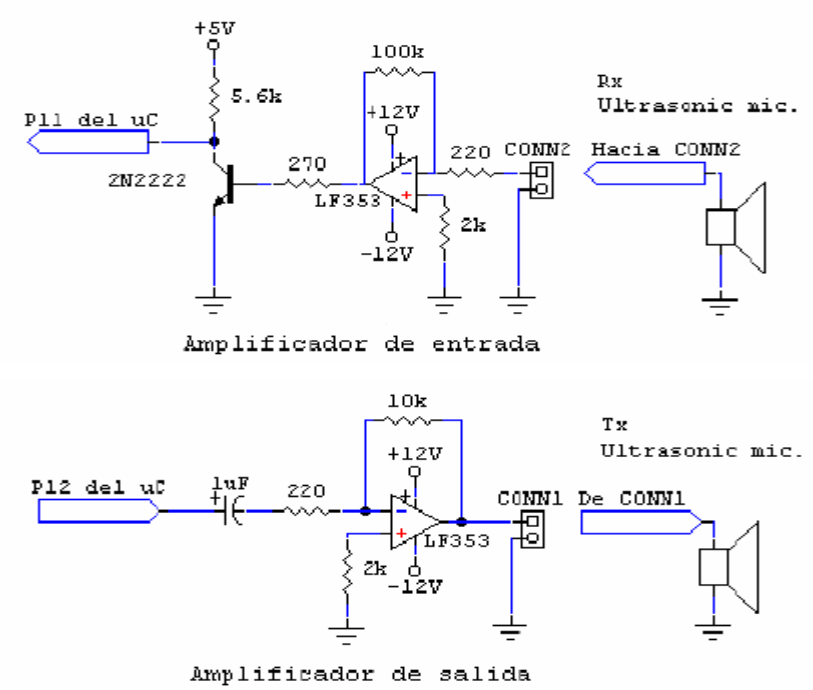
$$d_{recorrida} = \frac{346\,{}^{m}\!/_{S} * TCNT * 0.8\mu S}{2}$$

Reduciendo los términos tendríamos esta sencilla ecuación.

$$d_{recorrida} = 1384 * TCNT\mu m$$ *Ecuación No. 2*

Etapa de visualización: Esta etapa funcional se encarga de visualizar los resultados en el LCD. Para la visualización en el LCD el microcontrolador toma los resultados binarios, los convierte en decimales y luego en códigos ASCII, de aquí son enviados al puerto del microcontrolador en donde intervienen las rutinas de manejo del LCD.

Figura 21. Etapas de acondicionamiento de los sensores.



La fuente de alimentación para el medidor de ultrasonido es una fuente dual de +12 0 –12 con la que se alimentan el integrado LF353 y el sistema de desarrollo Motorola. La fuente fue montada con un par de reguladores LM7812 (positivo y negativo), un transformador dual de 9V 0V 9V, un puente rectificador y un par de filtros de 1000µF x 16V.

8.5.2 Diseño de la Válvula Inteligente

Este elemento permitirá hacer la manipulación remota y automática del actuador que en ultimas es una válvula de ½" conocida como llave de paso de bola, esta es conectada a una piñonearía encargada de dar un mayor troqué en el sistema mecánico que mueve la válvula, a través de un motor de paso que tiene una relación de vueltas de 1,2 grados por paso, este a su vez es manejado por circuito de potencia, usando para ello cuatro transistores TIP29 que tienen características de voltaje y corriente necesarias para la operación del motor (ver figura No. 23).

En el diseño del programa de manipulación del motor paso a paso, se hace indispensable manejar una secuencia específica en los bobinados del motor, para poder obtener la dirección de giro y el torque deseado, este motor cuenta con cuatro bobinas (A, B, C, D). Para obtener el mayor torque posible se están energizando dos bobinas de forma simultánea, logrando también que los pasos del motor sean más pequeños.

Antes de empezar a mover el motor se requiere que se tengan energizadas dos bobinas (A, B), dado que esta inicialización solo se realiza una única vez, este proceso fue implementado en Software.

La etapa de potencia que esta conformada por cuatro pares darlinton manejados por sus respectivos buffers 74125, como se puede ver en el esquema de la figura No. 22, esta a su vez es controlada por cuatro pines del microcontrolador que pertenecen a los puertos D4 a D0. Esto hace posible la manipulación del motor por medio un programa implementado en lenguaje ensamblador para microcontroladores de la **CPU08** de Motorota. Finalmente podemos manipular la válvula de manera digital, cambiando un registro de 8 bits e invocando una rutina de control ya desarrollada en el MC68HC908JK3, con el fin de posicionar la válvula en valores entre el 0% y el 100% de apertura.

Figura 22. Esquema de la etapa de potencia del motor Paso Paso.

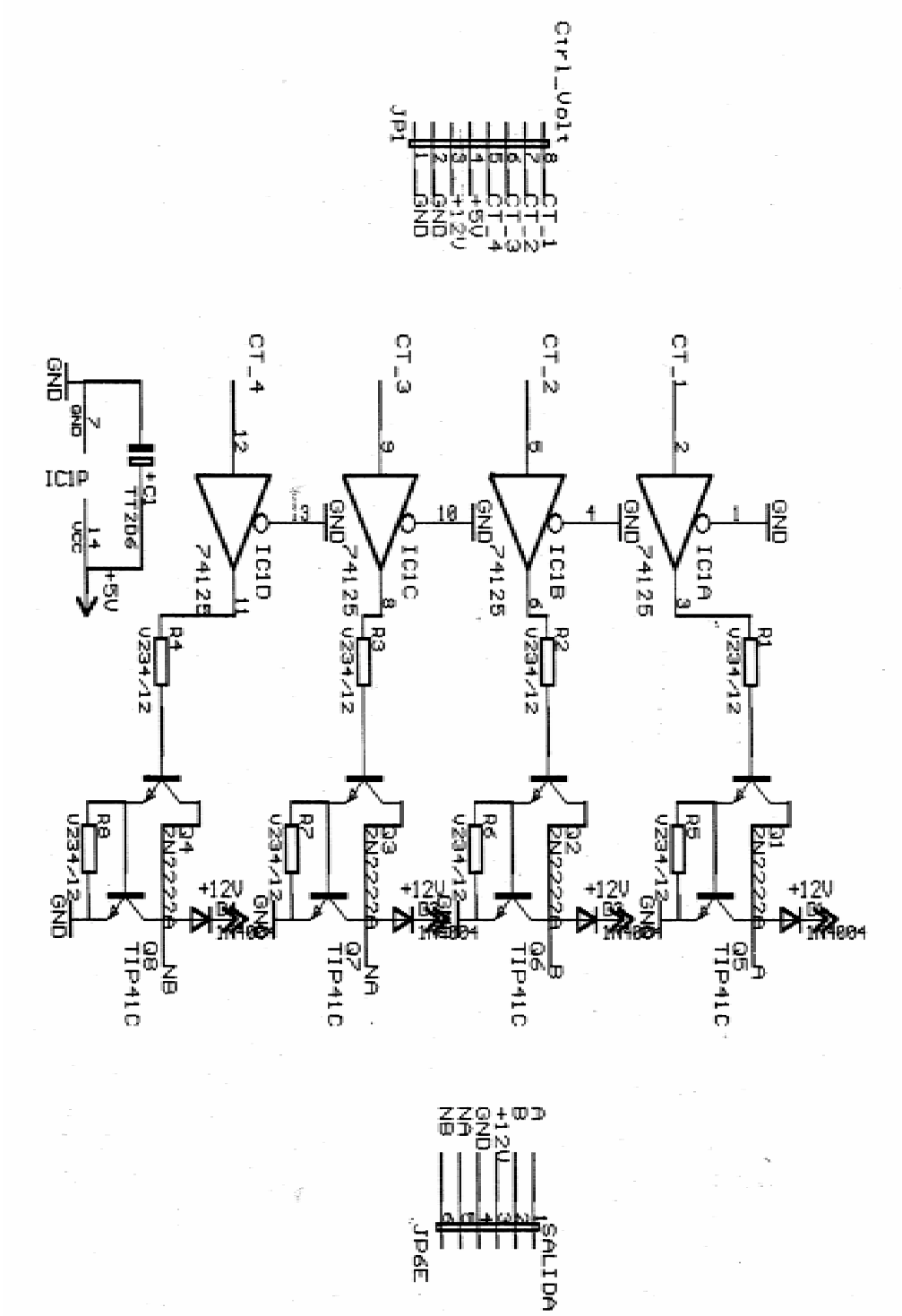
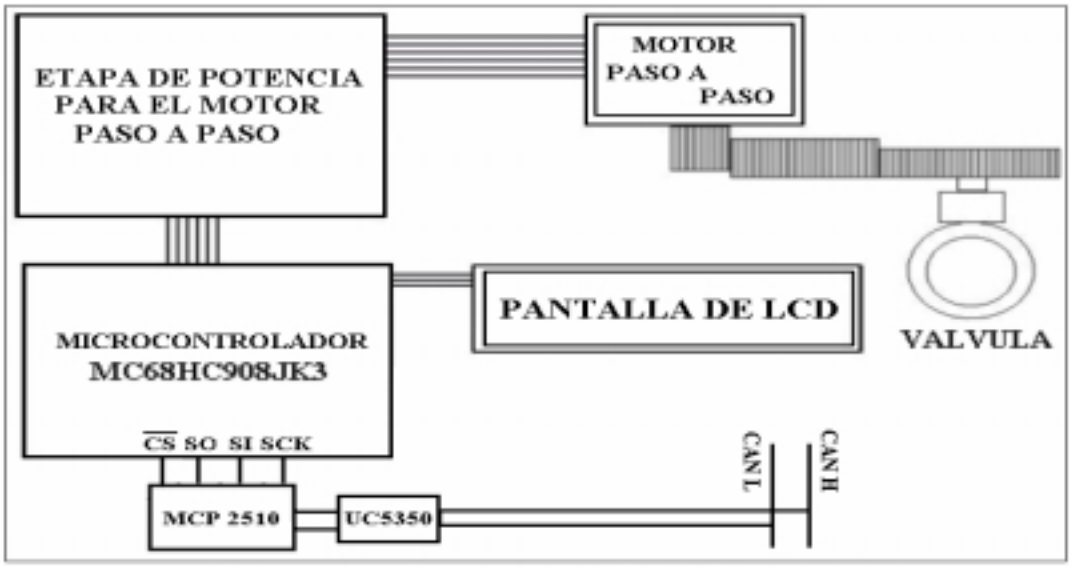


Figura 23. Diagrama de la Válvula Inteligente.



Otra etapa importante de este modulo es la etapa de comunicación que es la encargada de enviar o recibir la información de la computadora que monitorea y envía a su vez el valor de referencia o nivel requerido en el tanque No. 1 (Ver Figura No. 23).

Por Ultimo se implementará un sistema de ***CONTROL DIFUSO*** para que el nivel del tanque permanezca en valor requerido por el operador sin que importe las variaciones a la entrada. Para esto es necesario estudiar la teoría de control difuso, lo cual se puede ver a continuación.

8.5.2.1 Teoría de Control Difuso

8.5.2.1.1 Lógica Difusa (Fuzzy Logic)

Lógica Convencional = Define la realidad en grados de verdad absolutos (0's ó 1's). Es rígida.

Lógica Difusa = Define la realidad en diferentes grados de verdad. Sigue patrones de razonamiento similar a los del pensamiento humano. Es Flexible.

Ejemplo: Según la lógica Convencional, un recinto está solamente "Oscuro" (0) ó "Claro" (1). Para la Lógica Difusa Son posibles condiciones intermedias como "muy Claro", "Algo Oscuro", "Ligeramente claro", etc.

La lógica Difusa nos permite ser relativamente "imprecisos" en la representación de un problema y aún así llegar a una muy buena solución.

La lógica formal es solo un caso particular de la Lógica Difusa.

1.1.1.1 La Lógica Difusa maneja la incertidumbre y la imprecisión.

Un ejemplo clásico es el de "aparcar un coche". Este se puede aparcar fácilmente cuando no se ha fijado una posición final exacta. Pero si se fija una posición final exacta, la solución de este problema tardaría mucho (cálculos complejos) y su costo sería elevado.

"Cuándo la COMPLEJIDAD de un problema crece, la posibilidad de analizarlo en términos PRECISOS disminuye ".

8.5.2.1.2 Breve Historia De La Lógica Difusa

- En los años 30's, Lukasiewicz Define la Lógica Multivaluada, como generalización de

su lógica trivaluada (0, ½, 1).

- En 1964, La noción de Conjunto Fuzzy aparece por primera vez en un memorándum de la universidad de California en Berkeley y es debida al ucraniano nacionalizado americano, Lofti Zadeh.
- En 1965, la revista *"Information and Control"* publica el memorándum anterior, en donde aparece el artículo de Zadeh, *"Fuzzy Sets"*.
- En 1971, Zadeh publica el artículo, "Quantitative Fuzzy Semantics", en donde Introduce los elementos formales que acabarían componiendo el cuerpo de la doctrina de la lógica Difusa y sus aplicaciones tal como se conocen en la actualidad.
- En 1974, el Británico Ebrahim Mandani, Demuestra la aplicabilidad de la lógica difusa en el campo del control. Desarrolla el primer sistema de control Fuzzy práctico, la regulación de un motor de vapor.
- A finales de los 70's, Los ingenieros daneses Lauritz Peter Holmbland y jens-Jurgen Ostergaard desarrollan el primer sistema de control difuso comercial, destinado a una planta de cemento.
- Los japoneses empiezan a explotar la lógica difusa de forma masiva. Los occidentales asumieron una actitud reacia principalmente por dos razones: la primera era porque la palabra "Fuzzy" sugería algo confuso y sin forma, y la segunda porque no había forma de probar analíticamente que la teoría funcionaba correctamente, ya que el control fuzzy no estaba basado en modelos matemáticos.
- Aparecen toda una serie de investigadores japoneses en el campo de la lógica difusa tales como Sugeno, Togai, Bart Kosko (el fuzzsensei), entre otros.
- En 1986, Yamakawa, publica el articulo, "Fuzzy Controller hardward system". Desarrolla controladores fuzzy en circuitos integrados.
- En 1987, se inaugura en Japón el subterraneo de Sendai, uno de los más espectaculares sistemas de control difuso creados por el hombre. Desde entonces el controlador inteligente ha mantenido los trenes rodando eficientemente.
- En 1987, "FUZZY BOOM", se comercializan multitud de productos basados en la lógica difusa (sobre todo en Japón).

Productos basados en la Lógica Difusa.

- Cámaras Fotográficas y de Vídeo
- Sistemas de reconocimiento de voz
- Electrodomésticos.
- Alarmas.
- Controladores industriales.
- Robótica.
- Reactores nucleares, para mejorar la seguridad.
- Dispositivos médicos, y otros sistemas relativamente complejos.

Dada la facilidad de la lógica difusa para representar conocimientos, se ha empleado también en la solución de problemas sociológicos, sicológicos, políticos, administrativos, económicos, epidemiológicos y de otras disciplinas. Existen paquetes como el Fuzzy Decision Maker, que ayudan a las personas a tomar decisiones de todo tipo, como por ejemplo solucionar un problema familiar.

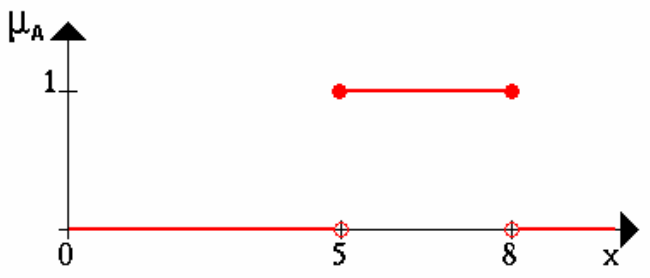
8.5.2.1.3 Conjunto Difuso (Fuzzy Set)

Conjunto Difuso: **Es un conjunto que puede contener elementos con grados parciales de pertenencia, a diferencia de los** Conjuntos Clásicos (Crisp Sets) **en los que los elementos pueden solamente "pertenecer" ó "No Pertenecer" a dichos conjuntos.**

Ejemplo 1.
Sea un conjunto *X*, formado por todos los números reales entre 0 y 10 (Universo de Discurso). Sea un conjunto *A* (subconjunto de X) definido como sigue, $A = \{x \mid 5 \leq x \leq 8\}$

El conjunto *A* se puede representar mediante su *Función de Pertenencia* (Función que asigna a cada elemento de *X* [1] ó [0] dependiendo si el elemento pertenece o no pertenece al conjunto) como se ilustra en la siguiente figura:

Figura 24. Representación Grafica de un Conjunto Difuso.



El conjunto *A* es un conjunto clásico (Crisp). Aunque este tipo de conjuntos tiene mucha aplicabilidad en distintas áreas, hay ocasiones en donde la falta de flexibilidad de estos conjuntos los hacen inapropiados, como se muestra a continuación.

Sea *B = {conjunto de la gente joven}*.

Un intento para construir este conjunto es definir un intervalo en años (conjunto clásico) de la siguiente manera:

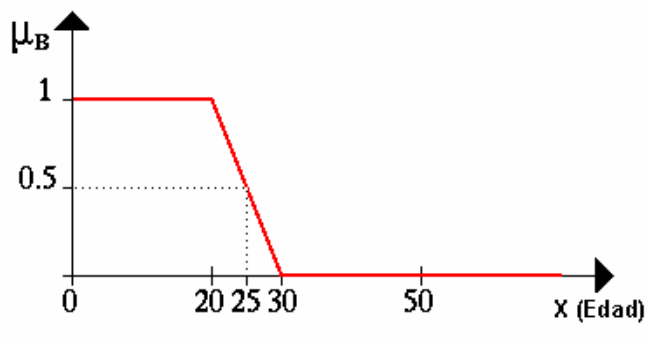
$$B = [\ 0\ ,\ 20\ ] = \{\ x \mid 0 \leq x \leq 20\ \}$$

Lo anterior implicaría que una persona sería joven hasta el día de su cumpleaños número 20, pero al siguiente día ya no lo sería. Ahora, si se cambiase el limite superior del intervalo el problema persistiría.

Una forma más natural de construir el conjunto *B*, es eliminando esa estricta separación entre ser joven y no serlo, admitiendo grados de pertenencia intermedios entre [0] y [1]. Por lo tanto el conjunto *B* será un Conjunto Difuso.

La función de pertenencia que describe el conjunto *B* sería la siguiente:

Figura 25. Representación del conjunto de la gente joven.



$$B = \left((x, \mu_B(x)), \begin{cases} \mu_B(x) = 1 & 0 \le x \le 20 \\ \mu_B(x) = -0.1x + 3 & 20 < x < 30 \\ \mu_B(x) = 0 & x \ge 30 \end{cases} \right)$$

De esta manera una persona de 25 años es todavia joven pero con un grado del 50%.

Función de Pertenencia o Membership Function (MF): Es una curva que determina el grado de pertenencia de los elementos de un conjunto. Se denota generalmente por μ y puede adoptar valores entre 0 y 1.

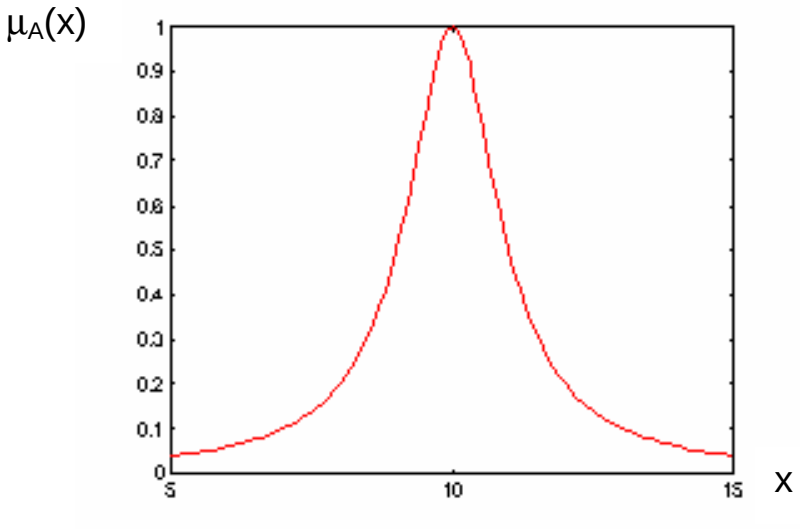
Universo de Discurso: Conjunto de valores que puede tomar una variable.

Ejemplo 2.

Sea el conjunto, A = Los números reales cercanos a 10. Ese conjunto podría estar representado de la siguiente forma:

$$A = \{ (x, \mu_A(x)) \ , \ \mu_A(x) = [1 + (x-10)^2]^{-1} \}$$

Figura 26. Representación del conjunto de los numeros reales cercanos a 10.



Nomeclatura: (No es una integral y sumatoria matemática)

$$A = \int_U \frac{\mu_A(x)}{x}$$

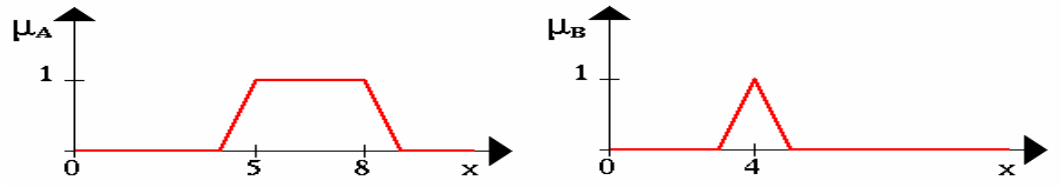
$$A = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i}$$

8.5.2.1.4 Operaciones Entre Conjuntos Difusos

Al igual que en los conjuntos clásicos, se define la intersección, la unión y el complemento para los conjuntos difusos.

Sean los conjuntos difusos *A* y *B* que se muestran en la siguiente figura:

Figura 27. Representación de los conjuntos A y B.

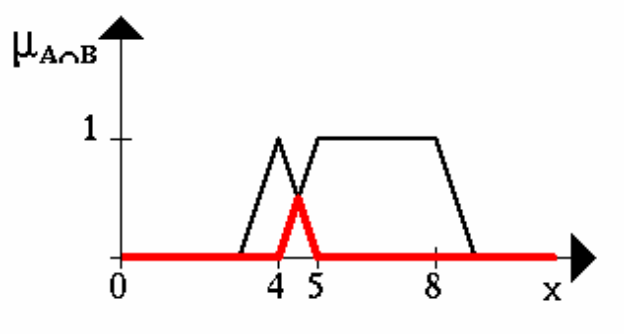


La intersección entre *A* y *B* se define de la siguiente manera:

$$C = A \cap B \ \ \forall x \in U$$

$$\mu_{A \cap B} = min\{\mu_A(x), \mu_B(x)\}$$

Figura 28. Representación de la intersección entre A y B.

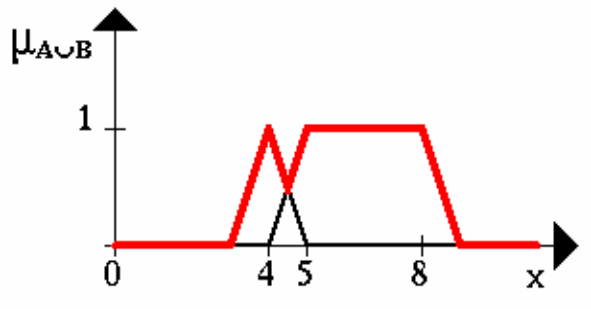


La unión entre *A* y *B* se define de la siguiente manera:

$$C = A \cup B \ \ \forall x \in U$$

$$\mu_{A \cup B} = max\{\mu_A(x), \mu_B(x)\}$$

Figura 29. Representación de la union entre A y B.

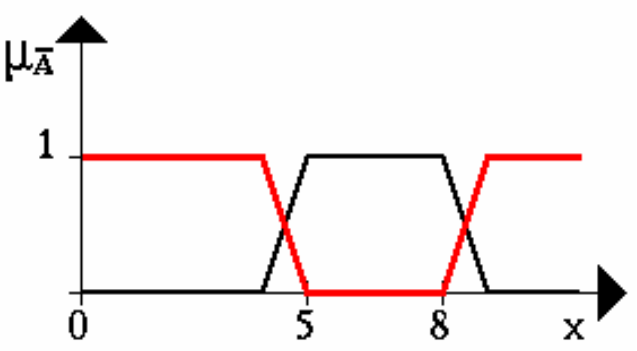


El complemento de un conjunto se define como sigue :

$$\mu_{\overline{A}} = 1 - \mu_A$$

Para el conjunto *A* se tendría,

Figura 30. Representación del complemento de A.



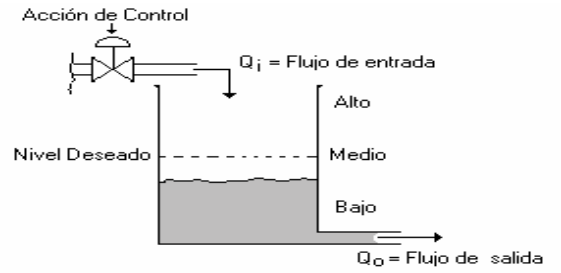
8.5.2.1.5 Control Difuso (Fuzzy Control)

En muchos procesos complejos, el control que ejerce un operador humano es más efectivo que el que proporciona un controlador automático convencional. Para esto el operador se basa en la experiencia (heurística) que tiene sobre el proceso.

El operador expresa sus estrategias de control Lingüísticamente como un conjunto de reglas de toma de decisiones.

Por ejemplo para un sistema de control de nivel de un tanque se podría tener,

Figura 31. Controlador de nivel Difuso.



“SI el nivel es muy bajo ENTONCES abra bastante la válvula”

“SI el nivel es bajo ENTONCES abra un poco la válvula”

“Si el nivel es medio ENTONCES no mueva la Válvula”

“SI el nivel es alto ENTONCES cierre un poco la válvula”

“SI el nivel es muy alto ENTONCES cierre bastante la válvula”

Donde las etiquetas lingüísticas como bajo, muy bajo, alto, bastante, etc. se modelan o definen mediante conjuntos difusos.

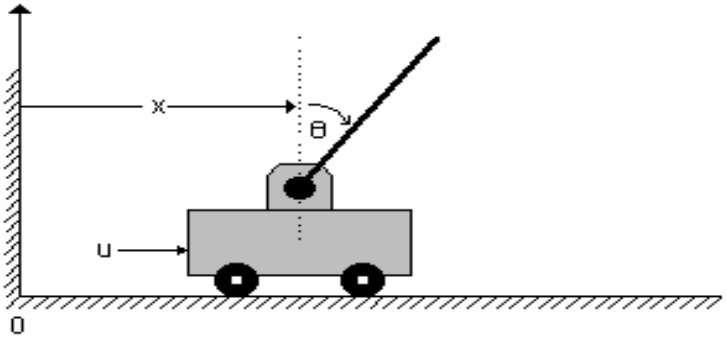
Si una variable puede tomar palabras en lenguaje natural (por ejemplo pequeño, rápido, etc.) como sus valores, esta variable se puede definir como una variable lingüística.

Para el ejemplo anterior, el nivel es una variable lingüística, así como la apertura de la válvula.

En síntesis y desde una perspectiva amplia, un controlador Difuso proporciona un algoritmo que puede convertir una estrategia de control lingüística, generalmente basada en la experiencia de un operador humano, en una estrategia de control automático.

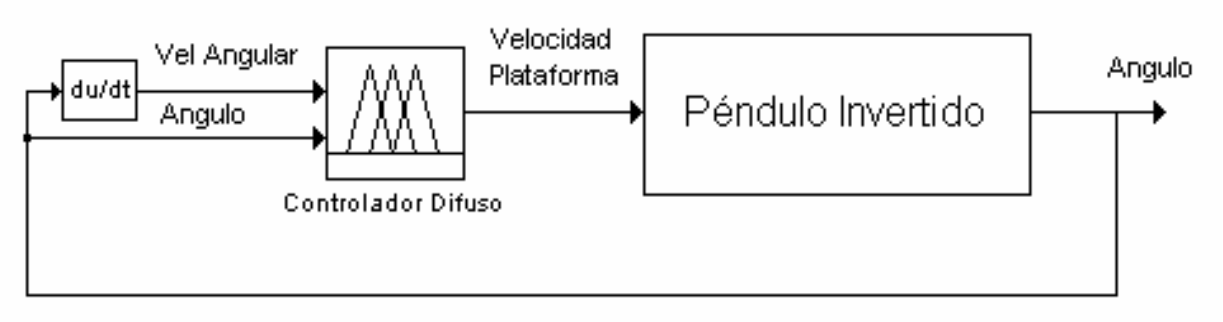
8.5.2.1.6 Ejemplo: control del péndulo invertido.

Figura 32. Sistema de Pendulo Invertido.



Problema: Mantener equilibrada una barra rígida sobre una plataforma movil que puede desplazarse en dos direcciones (Izquierda y Derecha).

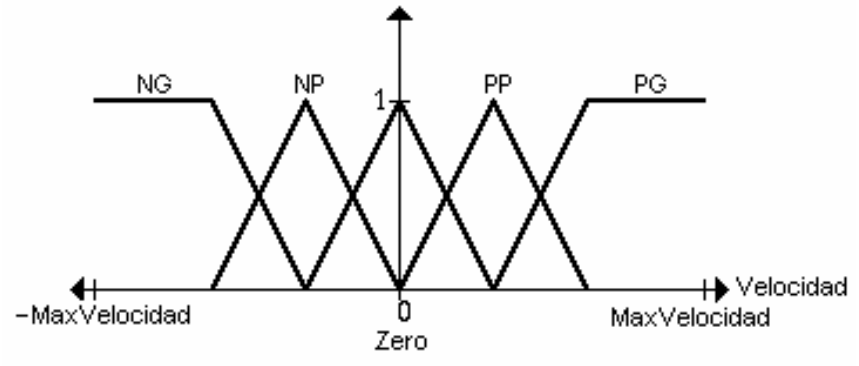
Figura 33. Digrama de bloques del Sistema de Control.



El controlador difuso tomará como entradas el angulo y la velocidad angular y como salida entregará la velocidad de la plataforma.

Primero, se definirán las diferentes etiquetas (alta, baja, etc.) de la variable lingüística velocidad (la de la plataforma):

Figura 34. Representación de la variable lingustica Velocidad de la Plataforma



Donde, NG = Negativo Grande, NP = Negativo Pequeño, Zero = Cero, PP = Positivo Pequeño, PG = Positivo Grande.

De igual manera se definen las funciones de pertenencia para el ángulo y la velocidad angular:

Figura 35. Representación de la variable lingustica Angulo.

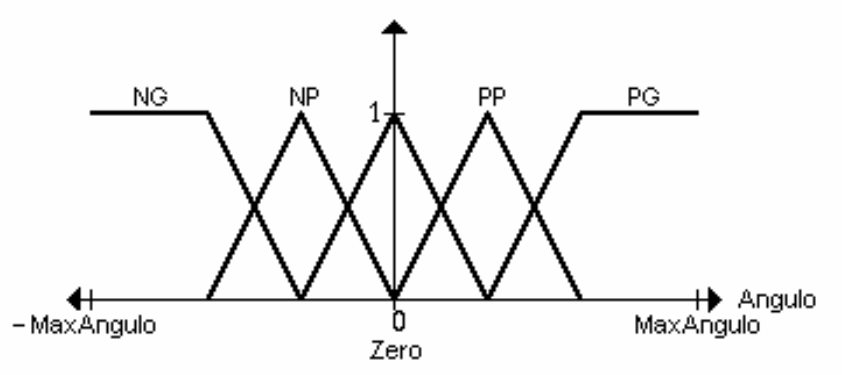
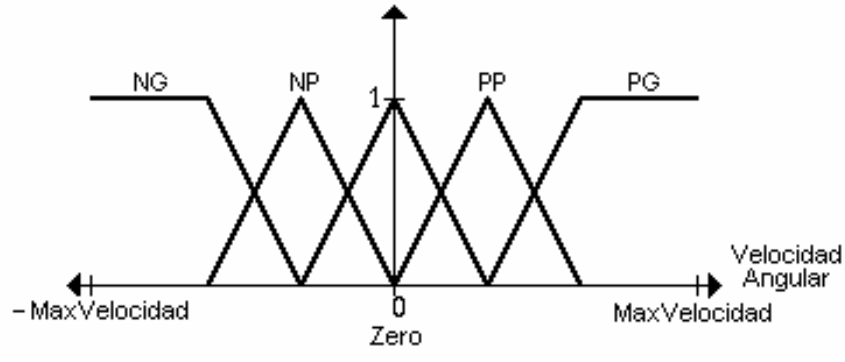


Figura 36. Representación de la variable lingustica de Velocidad Angular



Nota: Se asumirá que el ángulo inicial del pendulo esta muy cercano a posición de Equilibrio (Cero Grados).

La base de reglas de este controlador será la siguiente:

Tabla 6. Base de reglas para el controlador

VelAng \ Angulo	NG	NP	Z	PP	PG
NG			NG		
NP			NP	Z	
Z	NG	NP	Z (1)	PP	PG
PP		Z	PP (2)		
PG			PG		

La anterior tabla se interpretaría como sigue:

(1) Si (Angulo es Zero) y (Velocidad Angular es Zero) Entonces (Velocidad de la plataforma es Zero).

Esta es la situación en donde el péndulo está en equilibrio. La plataforma no se debe mover.

(2) Si (Angulo es Zero) y (Velocidad Angular es PP) Entonces (Velocidad de la plataforma será PP)

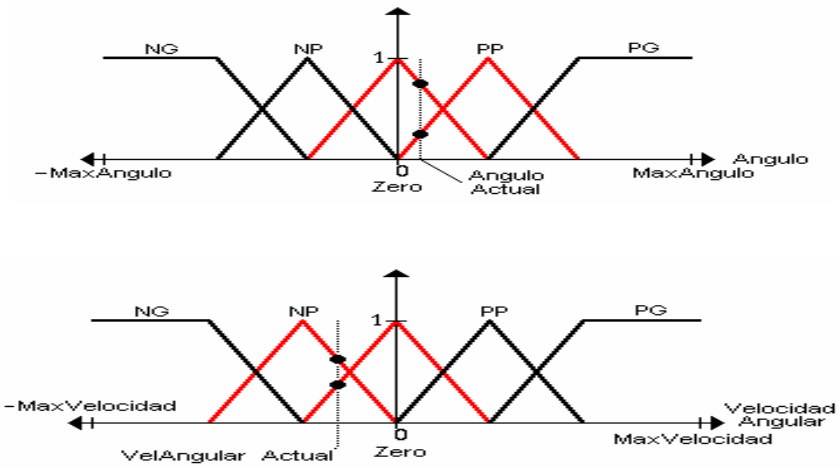
En este caso aunque el pendulo esta en la posición correcta se está moviendo lentamente en sentido positivo, por lo cual se hace necesario compensar este movimiento moviendo lentamente la plataforma en la misma dirección.

8.5.2.1.7 Proceso De Inferencia Difusa.

¿Como procesa la información de entrada (ángulo y velocidad angular) el controlador difuso para determinar la velocidad de la plataforma ?

Supogamos que en un momento dado se tiene un ángulo y una velocidad angular determinada, como se muestra en la figura:

Figura 37. Valores de Entrada de Angulo y Velocidad Angula.



Para estos valores de entrada se disparan 4 reglas. Estas son:

1. Si **(Angulo es Zero)** y **(Velocidad Angular es Zero)** Entonces **(Velocidad de la plataforma es Zero).**
2. Si **(Angulo es Zero)** y **(Velocidad Angular es NP)** Entonces **(Velocidad de la plataforma es NP).**
3. Si **(Angulo es PP)** y **(Velocidad Angular es Zero)** Entonces **(Velocidad de la plataforma es PP).**
4. Si **(Angulo es PP)** y **(Velocidad Angular es NP)** Entonces **(Velocidad de la plataforma es Zero).**

Evaluemos la primera regla, Si (Angulo es Zero) y (Velocidad Angular es Zero) Entonces (Velocidad de la plataforma es Zero).

Figura 38. Fuzzificacion de las variables de entrada.

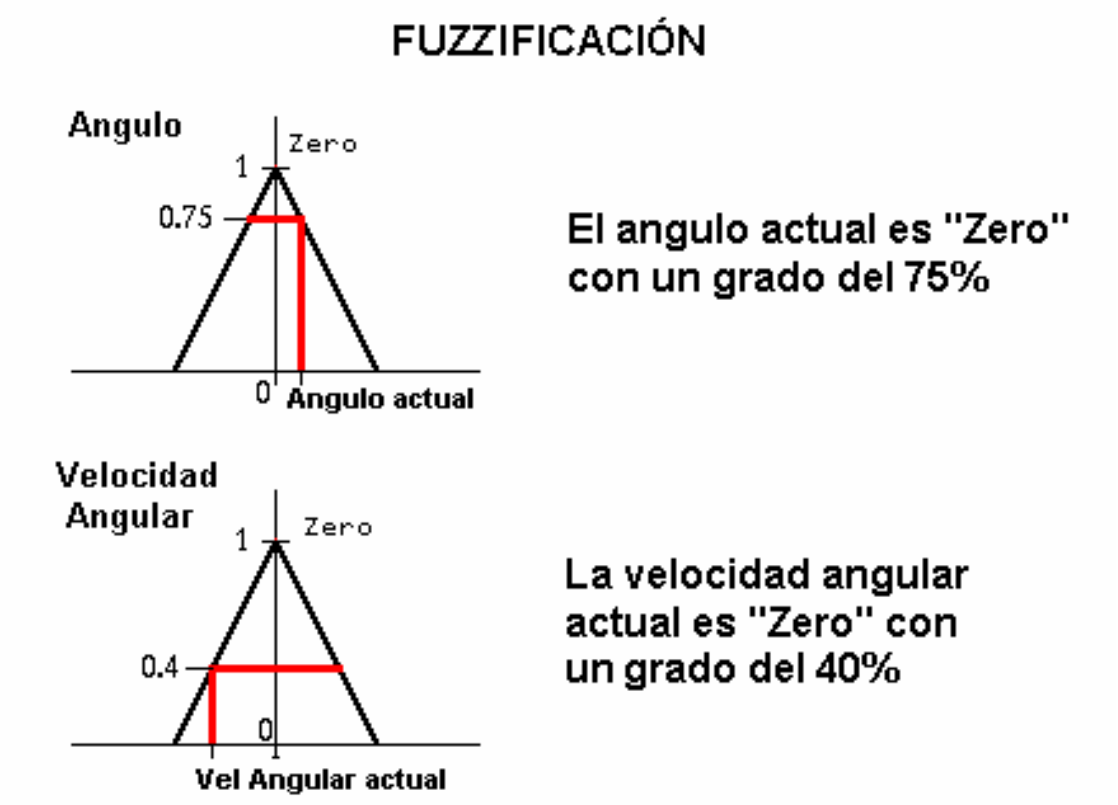
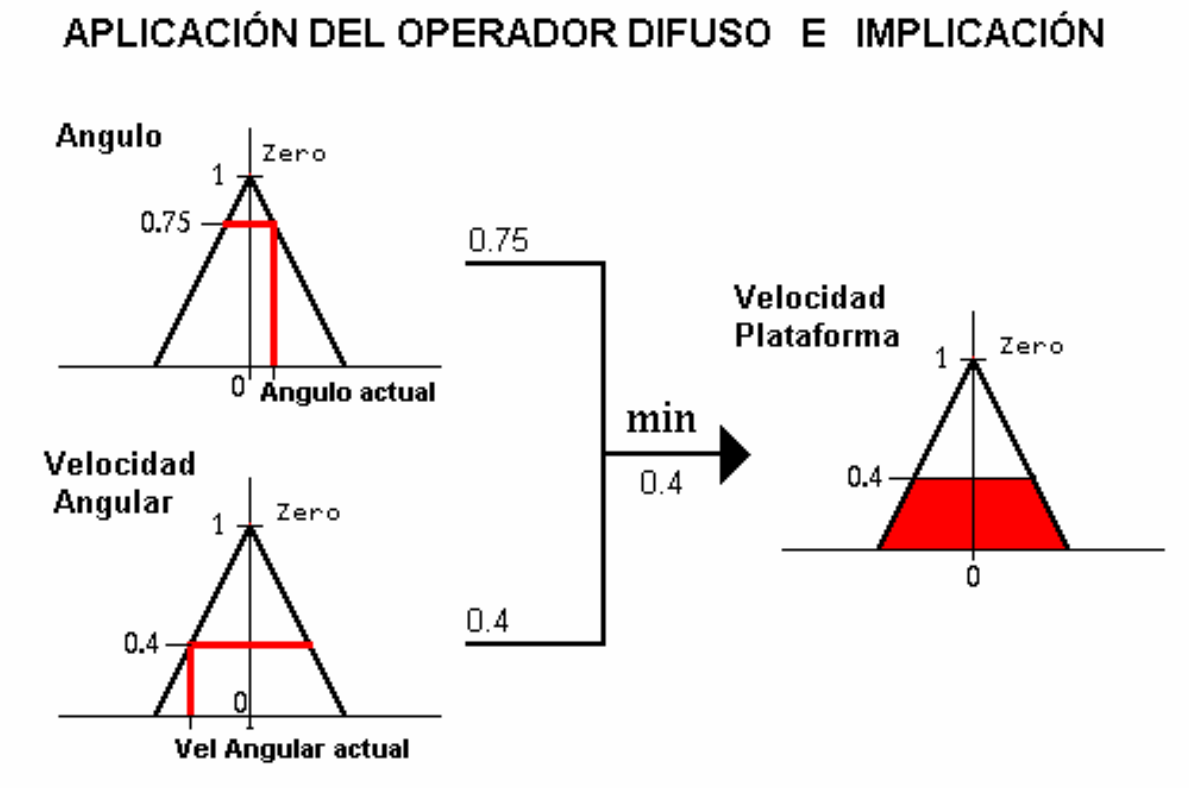


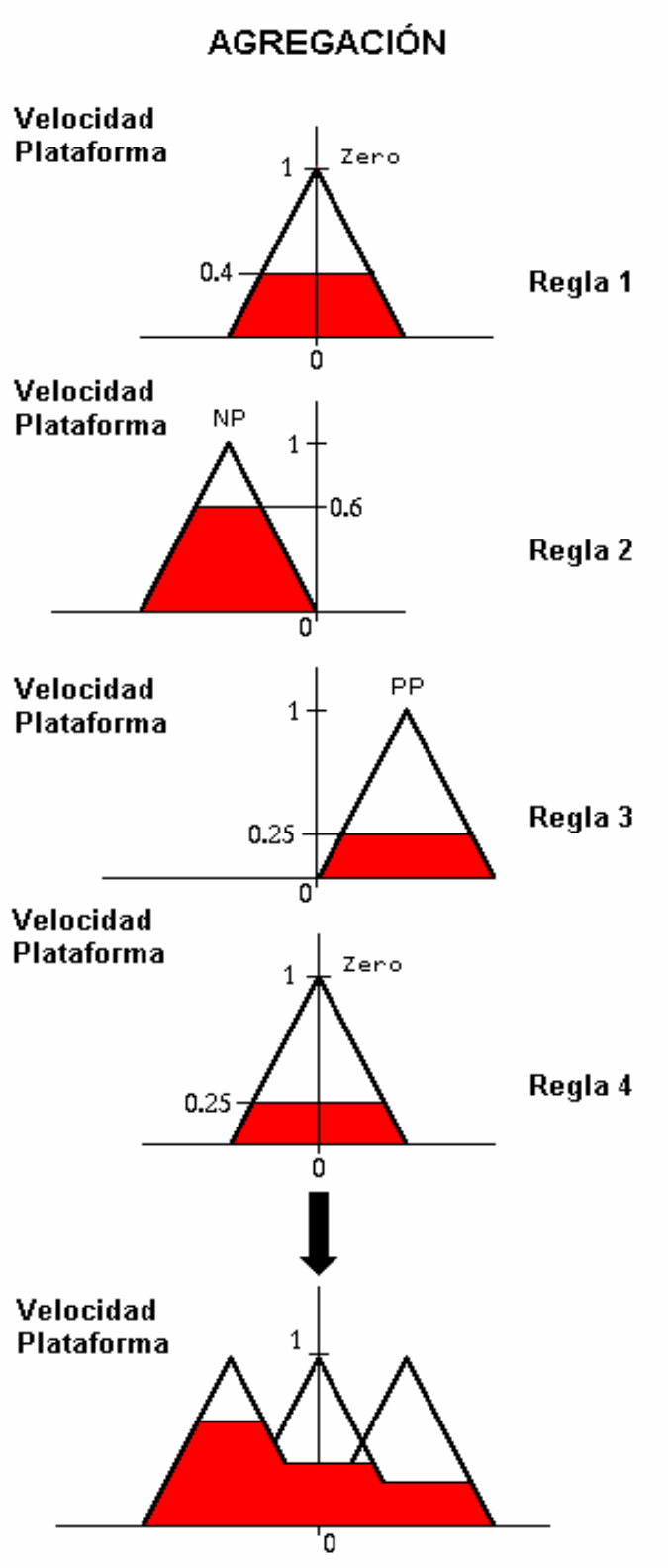
Figura 39. Grafico de operadores difusos y su aplicación.



Como el operador en este caso es “y”, se toma el mínimo de los valores que entrega el proceso de fuzzificación. Con este valor se corta a ese nivel (implicación difusa) el conjunto difuso “Zero” de la variable de salida.

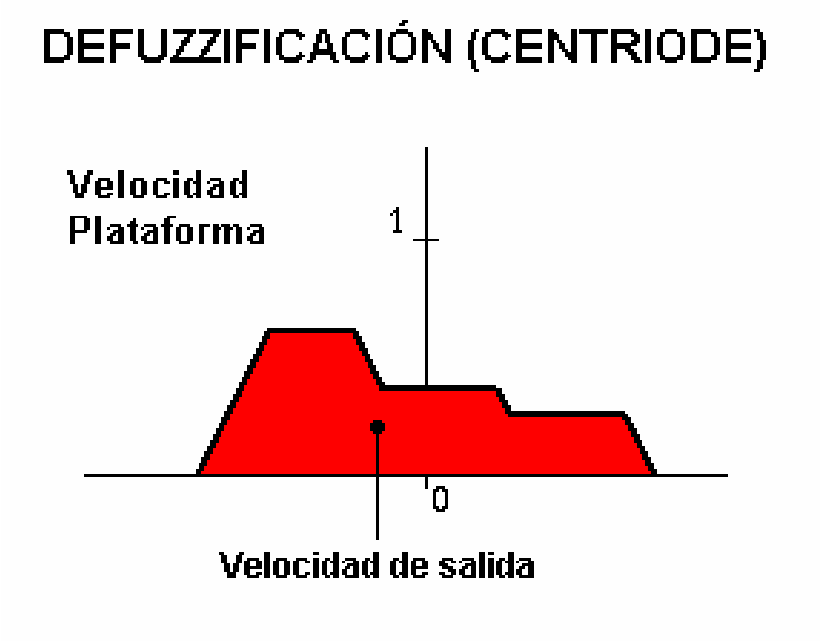
De igual manera se evaluan las otras tres reglas que se han disparado. Posteriormente se superponen los conjuntos difusos resultantes de cada regla para obtener un único conjunto de salida.

Figura 40. Grafico de Agragación de Reglas.



Hasta el momento el resultado que se tiene es un conjunto difuso de la variable de salida, por lo tanto se debe escoger un valor representativo de dicho conjunto para determinar la velocidad de la plataforma. Hay varios métodos (métodos de defuzzificación) entre ellos el de tomar el centro de masa del conjunto difuso:

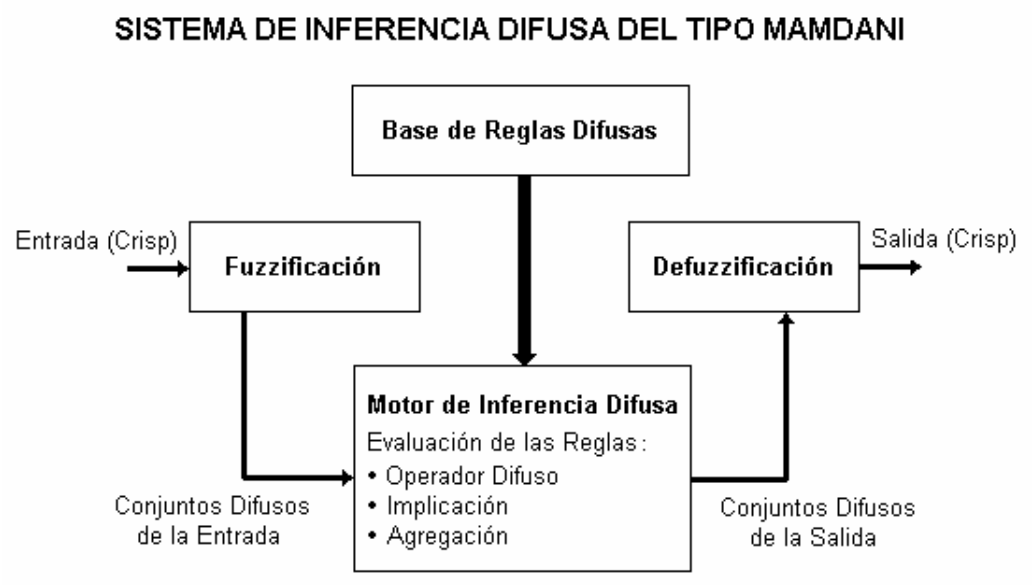
Figura 41. Velocidad de Salida de la Platafoma.



El sistema difuso presentado en este ejemplo se conoce como un sistema de tipo MAMDANI.

Finalmente, la estructura de un sistema de inferencia difusa de tipo Mamdani se puede observar en la figura siguiente:

Figura 42. Diagrama de Bloques de inferancia difusa tipo Mandani.

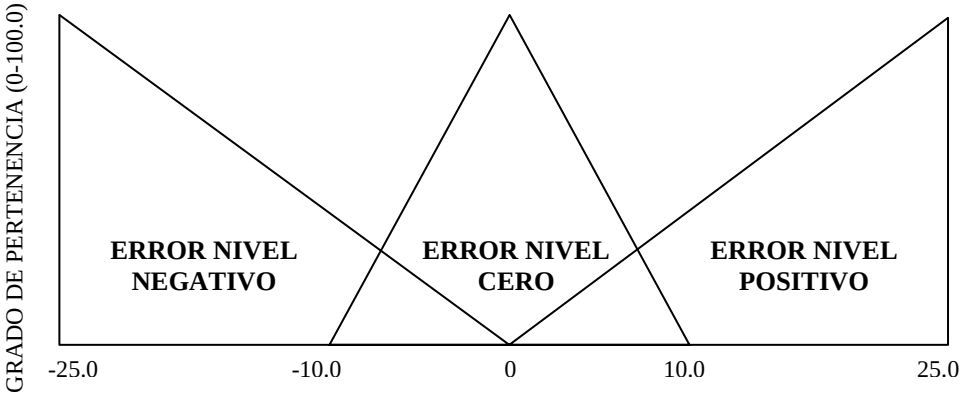


8.5.2.1.8 Situaciones en las cuales resulta benéfico emplear un controlador fuzzy

- Sistemas complejos que son difíciles o imposibles de modelar por métodos convencionales.
- Sistemas controlados por expertos Humanos.
- Sistemas que utilizan la observación humana como entrada o como base de las reglas.
- Sistemas que son confusos por naturaleza, como los encontrados en las ciencias sociales y del comportamiento.

8.5.2.2 Diseño del Controlador difuso para la Válvula

Figura 43. ERROR NIVEL = REREERENCIA – NIVEL ACTUAL



8.5.2.2.1 Calculo de ecuaciones de los grados de pertencia para ERROR NIVEL

Gp: GRADO DE PERTENENCIA
Err: ERROR NIVEL
m : PENDIENTE
b : Punto de Corte en Gp

[-25.0 A 0]

$$m = \frac{0-(100.0)}{0-(-25.0)} = -4$$ Gp(Err)=m(Err)+b ; b=0 | Gp(Err)= -4·Err

[-10.0 A 0]

$$m = \frac{100.0-0}{0-(-10.0)} = 10$$ Gp(Err)=m(Err)+b ; b=100.0 | Gp(Err)= 10·Err+100.0

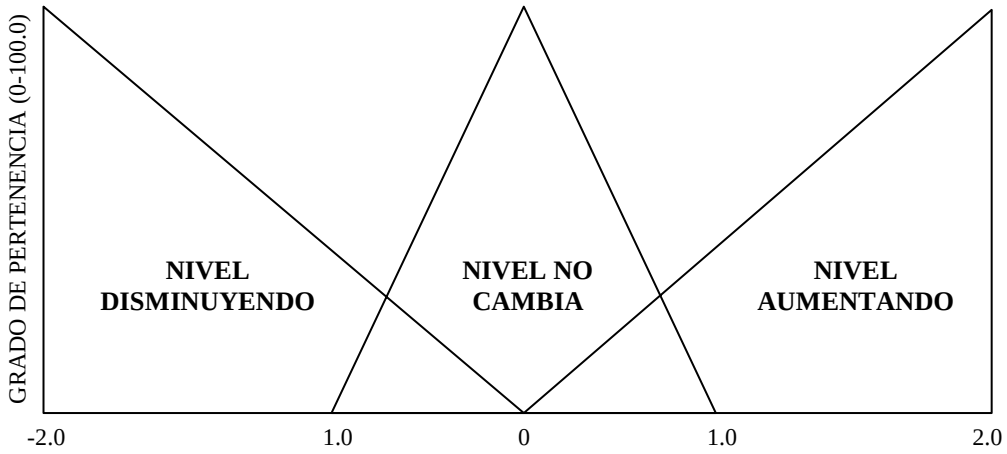
[0 A 10.0]

$$m = \frac{0-100.0}{10.0-0} = -10$$ Gp(Err)=m(Err)+b ; b=100.0 | Gp(Err)= -10·Err+100.0

[0 A 25.0]

$$m = \frac{100.0-0}{25.0-0} = 4$$ Gp(Err)=m(Err)+b ; b=0 | Gp(Err)= 4·Err

Figura 44. CAMBIO NIVEL = NIVEL ACTUAL – NIVEL ANTERIOR



8.5.2.2.2 Calculo de ecuaciones de los grados de pertencia para CAMBIO NIVEL

Gp: GRADO DE PERTENENCIA
Cnv: CAMBIO NIVEL
m : PENDIENTE
b : Punto de Corte en Gp

[-2.0 A 0]

$$m = \frac{0-(100.0)}{0-(-2.0)} = -50$$ Gp(Cnv)=m(Cnv)+b ; b=0 Gp(Cnv)= -50·Cnv

[-1.0 A 0]

$$m = \frac{100.0-0}{0-(-1.0)} = 100$$ Gp(Cnv)=m(Cnv)+b ; b=100.0 Gp(Cnv)= 100·Cnv+100.0

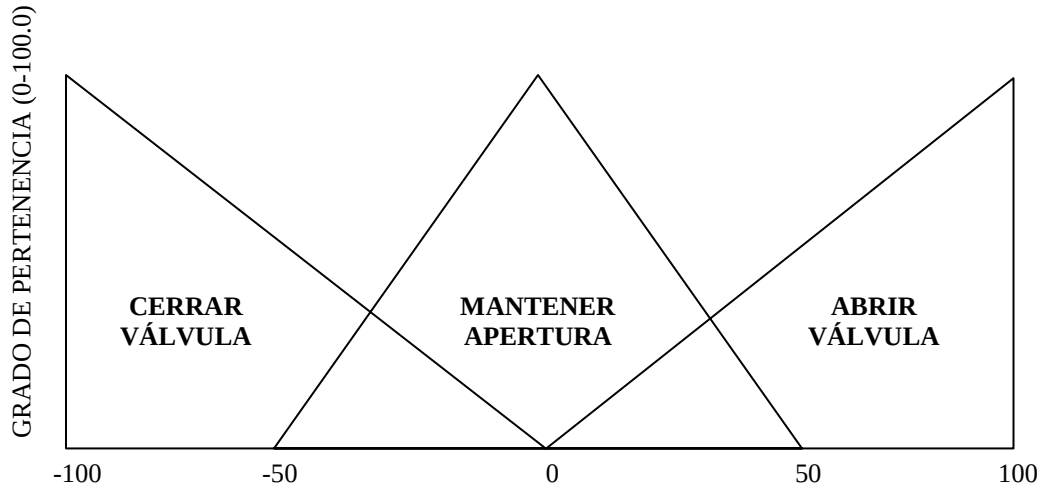
[0 A 1.0]

$$m = \frac{0-100.0}{1.0-0} = -100$$ Gp(Cnv)=m(Cnv)+b ; b=100.0 Gp(Cnv)= -100·Cnv+100.0

[0 A 2.0]

$$m = \frac{100.0-0}{2.0-0} = 50$$ Gp(Cnv)=m(Cnv)+b ; b=0 Gp(Cnv)= 50·Cnv

Figura 45. ACCIÓN DE CONTROL (APERTURA)



8.5.2.2.3 Calculo de ecuaciones de las aperturas (funciones inversas)

Gp: GRADO DE PERTENENCIA
Apt: APERTURA
m : PENDIENTE
b : Punto de Corte en Apt

[-100 A 0]

$$m = \frac{0-(100.0)}{0-(-100)} = -1$$ Apt(Gp)=m(Gp)+b ; b=0 Apt(Gp)= -Gp

[50 A 0]

$$m = \frac{0-(-50)}{100.0-0} = 0.5$$ Apt(Gp)=m(Gp)+b ; b=-50 Apt(Gp)= 0.5·Gp-50

[0 A 50]

$$m = \frac{50-0}{0-100.0} = -0.5$$ Apt(Gp)=m(Gp)+b ; b= 50 Apt(Gp)= -0.5·Gp+50

[0 A 100]

$$m = \frac{100-0}{0-100.0} = 1$$ Apt(Gp)=m(Gp)+b ; b=0 Apt(Gp)= Gp

8.5.2.2.4 Base de reglas para el controlador difuso de nivel en un tanque

En la tabla No. 7 se puede ver que sea han asignado en un registro los de 8 bits los posibles 8 estados en que pueden entrar las variables de Error Nivel y Cambio Nivel. Esto ayuda a hacer la evaluación de las reglas de control que están en la tabla No. 8. Esto se logra gracias a una serie de comparaciones hechas en el microntrolador entre los valores del Nivel Actual y el Nivel Anterior, para el caso de la variable Cambio Nivel y Referencia y Nivel Actual para el caso de la variable del Error Nivel.

En la tabla No. 9 y No. 10 se pueden ver las ecuaciones halladas desde las Figuras No. 43 y No. 44 a partir de las cuales se puede hacer la fuzificación de las variables, es decir asignar los valores de entrada a cada uno de los conjuntos borrosos de entrada establecido las figuras mencionadas. Los conjuntos borrosos de salida esta determinados entonces en la tabla de correspondencia de estados (Tabla No. 11), la cual fue hallada de la figura No. 45 en la que se observa los rangos en los cuales cada ecuación es valida.

CAMBIO NIVEL = NIVEL ACTUAL – NIVEL ANTERIOR

ERROR NIVEL = RERERENCIA – NIVEL ACTUAL

Tabla 7. Registro de condiciones de 8 bits para implementación en ensamblador.

BIT	CONDICIÓN VALIDA REPRESENTADA	ESTADO	VARIABLE
B0	MAYOR QUE NIVEL ACTUAL	*DISMINUYENDO*	*CAMBIO NIVEL*
B1	MENOR O IGUAL QUE NIVEL ACTUAL	*AUMENTANDO*	
B2	MAYOR O IGUAL QUE EL TOPE (**-10.00000**)	*NO CAMBIA*	
B3	MENOR O IGUAL QUE EL TOPE (**10.00000**)		
B4	MAYOR QUE REFERENCIA	*NEGATIVO*	*ERROR NIVEL*
B5	MENOR O IGUAL QUE REFERENCIA	*POSITIVO*	
B6	MAYOR O IGUAL QUE TOPE (**-10.00000**)	*CERO*	
B7	MENOR O IGUAL QUE TOPE (**10.00000**)		

Tabla 8. Base de reglas en lenguaje natural y los respectivos bists activos.

VALOR REGISTRO			ERROR NIVEL	CONDICION	CAMBIO NIVEL	APERTURA
17d	11h	SI	*NEGATIVO* [**B4**]	Y	*DISMINUYENDO* [**B0**]	CERRAR
193d	C1h		*CERO* [**B6B7**]			MANTENER
33d	21h		*POSITIVO* [**B5**]			ABRIR
28d	1Ch		*NEGATIVO* [**B4**]		*NO CAMBIA* [**B2B3**]	CERRAR
204d	CCh		*CERO* [**B6B7**]			MANTENER
44d	2Ch		*POSITIVO* [**B5**]			ABRIR
18d	12h		*NEGATIVO* [**B4**]		*AUMENTANDO* [**B1**]	CERRAR
194d	C2h		*CERO* [**B6B7**]			MANTENER
32d	22h		*POSITIVO* [**B5**]			ABRIR

Tabla 9. Correspondencia de los estados a las ecuaciones en la entrada error nivel.

BITS ACTIVOS	ESTADO	ECUACIÓN A APLICAR	RANGO
B4	*NEGATIVO*	Gp(Err)= -4·Err	[-25.0 A 0]
B4B6	*CERO*	Gp(Err)= 10·Err+100.0	[-10.0 A 0]
B5B7	*CERO*	Gp(Err)= -10·Err+100.0	[0 A 10.0]
B5	*POSITIVO*	Gp(Err)= 4·Err	[0 A 25.0]

Tabla 10. Correspondencia de los estados a las ecuaciones en la entrada cambio nivel.

BITS ACTIVOS	ESTADO	ECUACIÓN A APLICAR	RANGO
B0	*DISMINUYENDO*	Gp(Cnv)= -50· Cnv	[-2.0 A 0]
B0B2	*NO CAMBIA*	Gp(Cnv)= 100· Cnv +100.0	[-1.0 A 0]
B1B3	*NO CAMBIA*	Gp(Cnv)= -100· Cnv +100.0	[0 A 1.0]
B1	*AUMENTANDO*	Gp(Cnv)= 50· Cnv	[0 A 2.0]

Tabla 11. Correspondencia de los estados a las ecuaciones en la salida apertura

ESTADO	ECUACIÓN A APLICAR	RANGO DE SALIDA
CERRAR	Apt(Gp)= -Gp	[-100 A 0]
MANTENER	Apt(Gp)= 0.5·Gp-50	[-50 A 0]
MANTENER	Apt(Gp)= -0.5·Gp+50	[0 A 50]
ABRIR	Apt(Gp)= Gp	[0 A 100]

8.5.2.3 Simulación en MATLAB de del Controlador Difuso

A continuación se muestra el diagrama de bloques (Figura No.46), en el que se muestra el sistema de control difuso y en la Figura No. 48 se observa la grafica de comportamiento del sistema a un escalón de 10 cm. En la Figura No. 47 se tiene el diagrama de bloques de la simulación hecha en MATLAB.

Figura 46. Diagrama de bloques del controlador difuso.

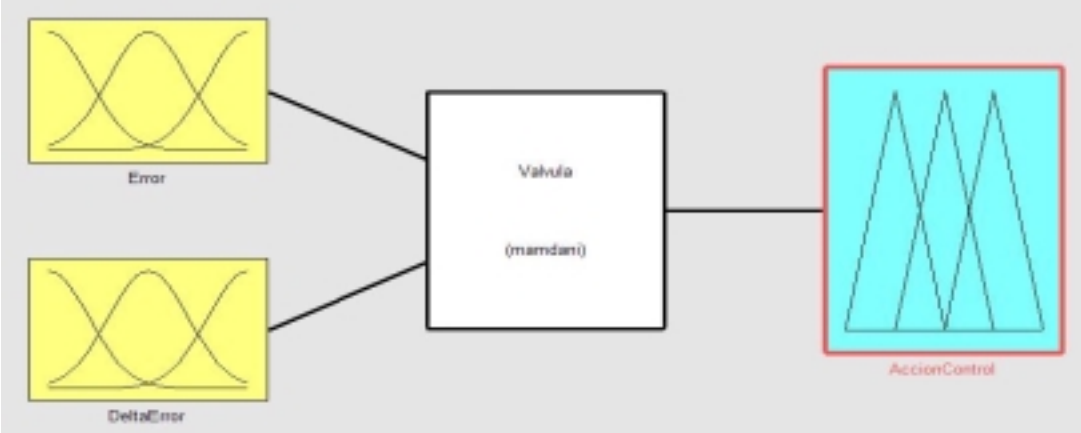


Figura 47. Diagrama de bloques de la simulación del controlador actuando en la planta.

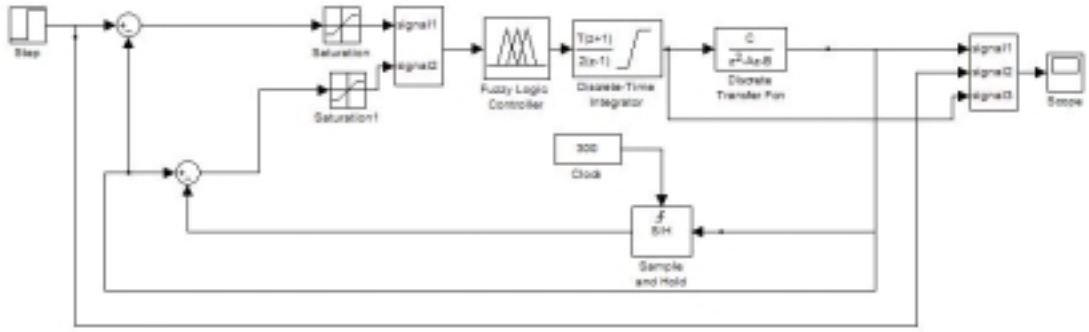
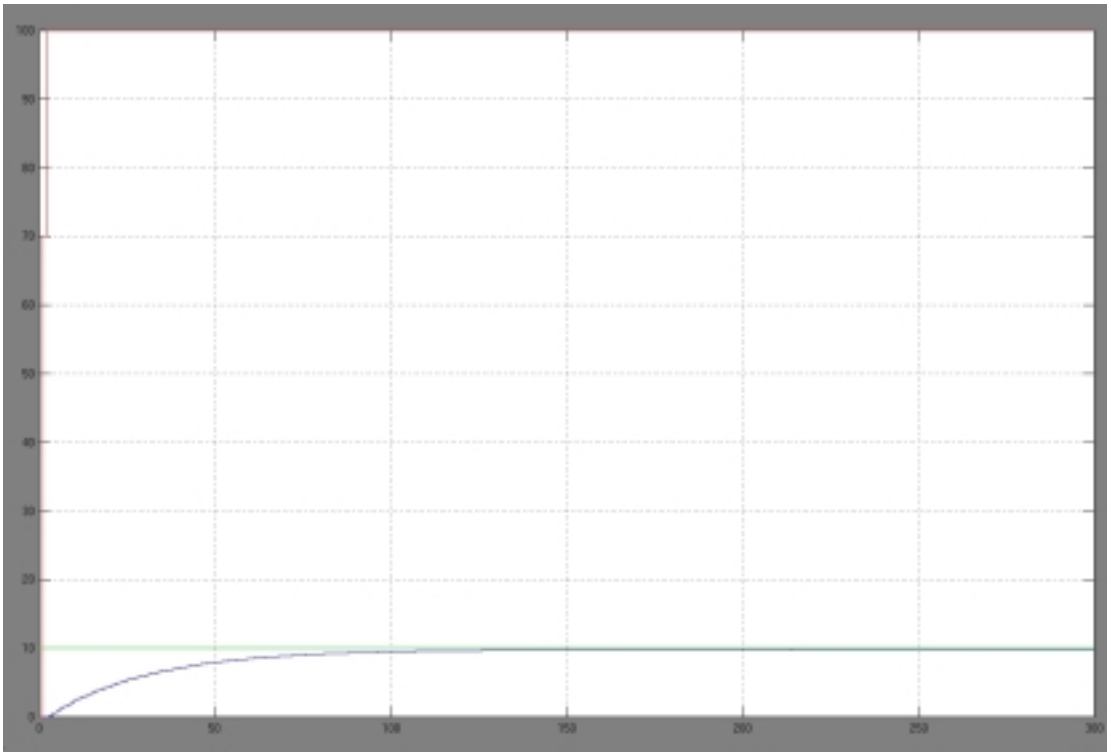


Figura 48. Grafico de la acción de control y la respuesta del sistema.

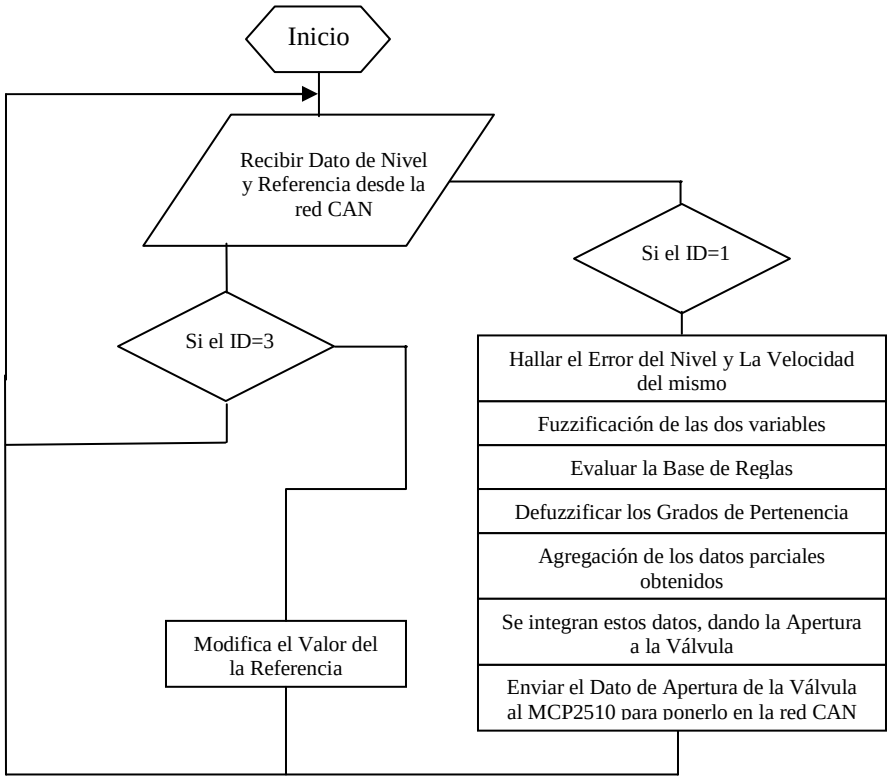


8.5.2.4 Conclusiones finales para el diseño del Controlador Difuso

Con la ayuda de las tablas Nos. 9, 10 y 11, se pueden establecer los rangos de entrada y de salida de cada conjunto difuso, con lo que se esta en capacidad de elaborar las rutinas necesarias para el diseño del controlador difuso que se requiere para la Válvula. Pues se hace una serie de comparaciones que nos llevan a determinar los bits del registro CONDICIONES Tabla No. 7, el cual puede ser consultado posteriormente para determinar que ecuaciones deben evaluarse, en la fuzzificación y en la desfuzzificación.

Teniendo las rutinas que resuelvan las ecuaciones de orden 1, que se han elaborado a partir de las Figuras 43, 44 y 45, el problema queda resuelto pues los resultados finales son agregados de tal manera que se tenga un valor o Acción de Control entre el -100% y el 100% que luego es sumada al valor anterior para obtener un valor de posición valido entre el 0% y 100%. En siguiente diagrama de flujo se observa la secuencia del programa.

Figura 49. Diagrama de Flujo del Programa para la Válvula Inteligente



8.6 DESARROLLO DEL SOFTWARE PARA LOS MICROCONTROLADORES QUE SE ENCARGARAN DE CONTROLAR LA VÁLVULA Y DE HACER LA MEDICIÓN DEL NIVEL.

Este software se ha diseñado en lenguaje ensamblador para la *CPU08* de *MOTOROLA (Ver anexos),* utilizando el compilador mencionado en sección 8.2.3. En los microcontroladores se desarrollaron los siguientes programas:

- Programa encargado de modificar los registros del MCP2510 para la comunicación de los módulos (asignación de mascaras y de identificadores).
- Programa de generación y captura de pulsos con operaciones enteras de 32 bits para medición del nivel en el tanque No.1 de la figura No. 17.
- Programa de manipulación de la válvula por medio de un compensador PI difuso.

8.7 DESARROLLO DEL SOFTWARE PARA EL MONITOREO Y CONFIGURACIÓN REMOTA DE LOS DISPOSITIVOS.

Este se diseño en Visual Basic de Microsoft con el uso de una librería de acceso dinámico (dll), desarrollada en Visual C++, la cual permite conectarnos a través de una interfase SPI desarrollado sobre un puerto ISA (***ver sección 8.1***) que a su vez se conecta con el nivel de Enlace de Datos de la red CAN.

En el programa se colocaron dos ventanas que grafican el valor de la variable de Nivel y el la acción de control ejecutada en la Válvula en términos de %, estos valores también se observan en dos cajas de texto debidamente etiquetadas. También hay un control que permite manipular el valor deseado de Nivel.

Esta aplicación consta de un programa Servidor y un programa Cliente, esto permite que se pueda monitorear y manipular el sistema desde el equipo Cliente conectado al Equipo

Servidor a través de una red LAN, pudiendo tener acceso al equipo donde se encuentre instalado el Sistema de Control de Nivel.

El programa con el código fuente se encuentra en los anexos al proyecto.

Advertencia: Todo el software desarrollado que se anexa al documento es con propósitos educativos y no comerciales. En caso de necesitar asesoria o ayuda sobre este proyecto enviar sus preguntas al autor en el correo electrónico velago@msn.com.

9 CONCLUSIONES

- El protocolo CAN es uno de los más usados actualmente en las industrias y otras aplicaciones críticas, esto es debido a su alta inmunidad al ruido y la baja probabilidad a no detectar errores en la transmisión o recepción de información entre los diversos nodos.

- En la actualidad el desarrollo de prototipos de sensores y actuadores para aplicaciones de control no es un gran problema, gracias a que ya se cuenta con microcontroladores y ASICs que implementan diversos protocolos.

- Los sistemas de control inteligente, permiten diseñar controladores capaces de hacer control de manera más confiable y flexible que los sistemas de control convencional, pues estos pueden llegar a manejar sistemas complejos con mucha precisión con capacidad aceptable para seguir funcionando a pesar de los cambios en la dinámica del sistema.

- Las nuevas herramientas de programación y desarrollo, tales como Visual Basic y los compiladores con simuladores y depuradores para microcontroladores, permiten hacer aplicaciones de bajo costo que son confiables a la hora de tener un producto final para su uso en las industrias.

- El diseño e implementación de controladores difusos en pequeños chips, pude llegar a ser complejo, pero las aplicaciones y las ventajas que tienen estos sistemas pueden hacer que en un futuro las industrias piensen en usar estos pequeños sistemas, como células inteligentes, capaces de comunicarse entre ellas y las estaciones de monitoreo, resolviendo los problemas en los diferentes procesos de control industrial.

BIBLIOGRAFÍA

AGUDELO, Oscar Mauricio. Introducción a la Lógica Difusa. Santiago Cali : Universidad Autónoma, 1999. 22 p.

CASTAÑO, Juan Andrés. Sistema de Desarrollo para Microcontrolador Motorola MC68HC908JK3/JK1. En : Electrónica & Computadoras N°63. (Abr. 2000); p. 18-24.

CREUS, Antonio. Instrumentación Industrial. 6 ed. Mexico : Alfaomega, 1998. 280 p.

HITACHI. HD44780U (LCD-II) Dot Matrix Liquid Crystal Display Controller/Driver. USA : HITACHI, 1998. 60 p.

KASCHEL, Héctor; PINTO, Ernesto. Análisis Protocolar del Bus de Campo CAN. Santiago : Universidad de Santiago de Chile, 2002. 5 p.

MICROCHIP. Smart Sensor CAN Node Using the MCP2510 and PIC16F876. USA : Microchip, 1999. 23 p.

--------. MCP2510 Rev. A Silicon Errata Sheet. USA : Microchip, 2000. 8 p.

--------. Stand-alone CAN Controller with SPI interface. USA : Microchip, 1999. 40 p.

--------. Controller Area Network (CAN) Basic. USA : Microchip, 1999. 180 p.

MOTOROLA. HCMOS Microcontroller Unit Technical Data. USA : Motorola, 1999. 210 p.

UNITRODE . UC5350 CAN Transceiver. USA : Unitrode, 2001. 15 p.

ANEXO A (Código fuente de los programas implementados en los microcontroladores)

```
*****************************************************
* Programa en ensamblador para Motorola CPU08
*****************************************************

RAMStart     EQU  $0080
RomStart     EQU  $EC00   ; Valid for all JL3, JK3, JK1
VectorStart  EQU  $FFDE

$Include 'jl3regs.inc'    ; For the 68HC908JL3, 68HC908JK3, 68HC908JK1
$include 'SPIEQU.inc"     ; Definicion de igualdades para el SPI.

   org RamStart

contador1 ds 1   ; Allows three timeout routines to be called each of which
CONTADOR  DS 1
wait1    ds 1
wait2    ds 1
wait3    ds 1 ;Contadores de Retardo.
instruc  ds 1 ;Bandera de Intruccion o Dato (LCD).
Dato     ds 1 ;a enviar al LCD.
RxCont   ds 1 ;Contador de Recepsion.
DLC      ds 1 ; Tamaño de datos a enviar.
;TxCont   ds 1 ;Contador de Transmision.
Bitscont ds 1 ;Contador de Bits para LCD (pto al 4094).
var      ds 1 ;Variable temporal para imprimir HEXA.
;var1     ds 2 ;Variable temporar para Guardar el H:X.

$include 'MAt.RAM'
   org RomStart
$include 'LCDRTS.ASM'   ; Rutinas del LCD, Impresion de cadenas y Conversiones
Frecuentes.
$include 'SPIRTS.ASM'   ; Rutinas de Lectura, Escritura, etc. del SPI.
$include 'NUMRTS.ASM'   ; Rutinas de Multiplicacion, Division, Resta y Visualizacion.
$include 'MCPRTS.ASM'   ; Rutinas del MCP2510.
*************************************************************
* Init_Timer - Esta rutina se encarga de iniciar el timer    *
*            para que envie los pulso a la salida del      *
*            timer 1 canal 0 (PTD4).                       *
*************************************************************
Init_Timer:
     mov   #$30,TSC     ; LIMPIA Y DETIENE EL CONTADOR, DIVISOR POR 1.
```

```
                ; frecuencia del bus = 1.25Mhz

    mov  #$00,TMODH   ; define la frecuencia de los pulsos de salida.
    mov  #$0E,TMODL   ;
    mov  #$14,tsc0    ; Interupciones deshabilitadas y Cambia valor del PTD4
                ; Cada vez que se alcanza el valor del TMOD.
    mov  #$00,tsc1    ; sin ninguna novedad.
    mov  #$40,TSC     ; Inicia el timer
    rts

***********************************
* INICIALIZACIÓN DEL TIMER
***********************************
INIT_TIM0
    MOV    #$36,TSC       ; LIMPIA Y DETIENE EL CONTADOR Y DIVISOR POR
64.
    MOV    #$FF,TCH0H
    MOV    #$FF,TCH0L     ; CARGA VALORES PARA RETARDO DE 1 SEG.
    MOV    #$70,TSC0      ; HABILITA INTERRUPCION DEL CANAL CERO.
    MOV    #$06,TSC       ; INICIA EL TIMER.
    RTS
*****************************************************************
* Main_Init - This is the point where code starts executing  *
*          after a RESET.                    *
*****************************************************************
Main_Init:
    rsp
    clra
    clrx
    jsr    INIT_SPI
    jsr    Init_MCP2510
    jsr    init_lcd
    jsr    clrlcd
    LDHX   #TITU
    JSR    SEND_CAD
    BSET   7,INSTRUC
    LDA    #$C0
    JSR    SEND
    BCLR   7,INSTRUC
    LDHX   #Tnivel
    JSR    SEND_CAD
***********************************
* INICIALIZACIÓN VARIABLES
***********************************
INIT:
```

```
      rsp
      clra
      clrx
      CLR A
      CLR B
      MOV    #$05,C       ;Constante de Multiplicacion para calcular la distancia
      MOV    #$68,D       ;Velocidad del sonido 340 m/s   ;1384
      CLR E
      CLR F
      CLR G
      CLR H
      CLR I
      CLR RES
      MOV     #$20T,CONTADOR
      MOV     #$10T,CONTADOR1
      BSR     INIT_TIMER
      CLI
main_loop
      bra main_loop
*****************************************************************
* IRQ1_ISR - IRQ1 Rutina de Interrupcion para              *
*          Recepcion de datos del MCP2510.              *
*****************************************************************
IRQ1_ISR:
       pshh
       ;jsr     print
       pulh
       rti
*****************************************************************
* TCH1_ISR - Timer Interrupt Service Routine.              *
*           after a RESET.                              *
*****************************************************************
TCH1_ISR:
       DBNZ   CONTADOR,CONTEO         ; Cuenta los pulso que retornan al sensor de
ultrasonido si conteo=20 entonce:
       CLR    TSC1                  ; Se desconecta la captura del timer.
       BSET   5,TSC                  ; Se detiene el timer quedando el tiempo de retardo en el
TCNT.
       MOV    TCNTH,A                ;
       MOV    TCNTL,B                ; Se almacena el dato en la variable AB.
      bset     7,instruc
      lda      #$02
      jsr      send
      bclr     7,instruc
```

```
    JSR     M16X16          ; Multiplicacion del tiempo almacenado y una constante de
16 bits.
    MOV     #$00,A
    MOV     #$26,B
    MOV     #$25,C
    MOV     #$A0,D          ; Tamaño del Tanque.
    JSR     RES32           ; Nivel del tanque = Tamaño del Tanque - Medicion de
distancia.
    lda     A
    CMP     #$FF
    BEQ     ESCERO
    BRA     NOESC
ESCERO
    MOV     #$0,A
    MOV     #$0,B
    MOV     #$0,C
    MOV     #$0,D
NOESC  lda     #$0
    ldx     #$31
    jsr     write_spi       ;Registro Parte alta Identificador
    lda     #$20
    ldx     #$32
    jsr     write_spi       ;Coloca el Identificador del Mensaje (Nivel Medido)
    lda     A
    ldx     #$36
    jsr     write_spi
    lda     B
    ldx     #$37
    jsr     write_spi
    lda     C
    ldx     #$38
    jsr     write_spi
    lda     D
    ldx     #$39
    jsr     write_spi   ; Coloca los Datos en los campos de datos
    lda     #RTS0
    jsr     rts_spi
    JSR     HEXBCD32        ; Visualizacion del Dato.
    sei                     ; deshabilita las interrupciones.
    ldx     #$20T
otra   jsr     espera2
    dbnzx    otra           ; Retado para hacer la proxima captura.
    jmp     INIT            ; reinicia el proceso.
CONTEO
    BCLR    7,TSC1          ; Se limpia la bandera de valor alcanzado.
```

```
      Rti

*****************************************************************
* T_ISR - Rutina de Atencion a la interrupcion del timer.    *
*****************************************************************
T_ISR:
   pshh
   DBNZ   CONTADOR1,PULSOS  ; Cuando el numero de pulsos sea 10 entonces:
    mov     #$00,TMODL      ; se limpia el registro modulo.
    MOV     #$00,TSC0       ; se detiene el tren la generacion de pulsos.
    MOV       #$00,TSC ; Inicializa el contador, tiempo con frecuencia base = 1.25Mhz.
     MOV     #$48,TSC1     ; se prepara el canal 1 para contar los pulsos que retornen.
                     ; se habilita la interrupcion y se captura en flanclo de bajada.
PULSOS
     BCLR      7,TSC         ; Limpia la bandera de valor alcanzando.
     pulh
     rti
*****************************************************************
* DUMMY_ISR - Dummy Interrupt Service Routine.              *
*          Just does a return from interrupt.           *
*****************************************************************
dummy_isr:
     rti         ; return
*****************************************************************
* Vectors - Timer Interrupt Service Routine.                *
*          after a RESET.                               *
*****************************************************************
  org  VectorStart
      dw  dummy_isr    ; ADC Conversion Complete Vector
      dw  dummy_isr    ; Keyboard Vector
      dw  dummy_isr    ; (No Vector Assigned $FFE2-$FFE3)
      dw  dummy_isr    ; (No Vector Assigned $FFE4-$FFE5)
      dw  dummy_isr    ; (No Vector Assigned $FFE6-$FFE7)
      dw  dummy_isr    ; (No Vector Assigned $FFE8-$FFE9)
      dw  dummy_isr    ; (No Vector Assigned $FFEA-$FFEB)
      dw  dummy_isr    ; (No Vector Assigned $FFEC-$FFED)
      dw  dummy_isr    ; (No Vector Assigned $FFEE-$FFEF)
      dw  dummy_isr    ; (No Vector Assigned $FFF0-$FFF1)
      dw  T_ISR       ; TIM1 Overflow Vector
      dw  TCH1_ISR    ; TIM1 Channel 1 Vector
      dw  DUMMY_ISR     ; TIM1 Channel 0 Vector
      dw  dummy_isr    ; (No Vector Assigned $FFF8-$FFF9)
      dw  IRQ1_ISR    ; ~IRQ1
      dw  dummy_isr    ; SWI Vector
      dw  main_init    ; Reset Vector
```

```
*****************************************************************************
*****
*          VALVULA INTELIGENTE CON COMUNICACION USANDO PROTOCOLO
CAN     *
* Este programa se encarga de realizar las funciones necesarias para hacer          *
* el control Difuso y  la comunicación con el protocolo CAN.                        *
* Hecho para una válvula de rotación mecánica, movida por un motor PASO A PASO      *
*****************************************************************************
*****
RAMStart     EQU  $0080   ; Origen de la RAM.

RomStart     EQU  $EC00   ; Origen del Programa en La Flash.
VectorStart  EQU  $FFDE   ; Ubicacion de los vectores de Interrupcion.

$Include 'jl3regs.inc'    ; Etiquetas de los Registros de configuracion 68HC908JK3-JL3-JK1
$include 'SPIEQU.inc''     ; Definicion de igualdades para el SPI.
   org RamStart

wait1    ds 1
wait2    ds 1
wait3    ds 1 ;Contadores de Retardo.
instruc  ds 1 ;Bandera de Intruccion o Dato (LCD).
Dato     ds 1 ;a enviar al LCD.
DLC      ds 1 ; Tamaño de datos recividos CAN
RxCont   ds 1 ;Contador de Recepsion.
TxCont   ds 1 ;Contador de Transmision.
Bitscont ds 1 ;Contador de Bits para LCD (pto al 4094).
var      ds 1 ;Variable temporal para imprimir HEXA.
var1     ds 2 ;Variable temporar para Guardar el H:X.
keys     ds 16 ;Espacio en memoria para almacenar datos recibidos en RX.

VALOR    DS 1 ;Valor BCD Tomado del Teclado.
DIGITO   DS 1 ;Digito Actual.
OPCION1  DS 1 ;Variable para menú
OPCION2  DS 1 ;Variable para menú
TECLA    DS 1 ;Dato capturado del teclado
NIVEL    DS 1 ;Nivel donde se encuentra el menú

VFINAL  DS  1 ;Posición final de la valvula (N de vueltas)
VACTUAL DS  1 ;Posición Actual de la valvula (N de vueltas)
VACTUAL_P DS  1 ;Posición Actual de la valvula (Porcentaje)

KCERO    DS  1 ;Constante de cero
Nvueltas ds  1 ;Numero de vueltas a dar por la valvula
KSPAN    DS  1 ;Constante de Span
```

```
SPAN     DS  1 ;Alcance

Referencia  ds 4
ErrorNivel  ds 4
CambioNivel ds 4
NivelTant     ds 4
NivelTact     ds 4

CONDICIONES    DS 1
GRADOVNC       DS 2     ; GRADO DE PERTENENCIA PARA LA VELOCIDAD (NO
HAY CAMBIO)
GRADOV         DS 2     ; GRADO DE PERTENENCIA PARA LA VELOCIDAD
(DISMINUYE) NEGATIVO                    ; O (AUMENTA) POSITIVO
GRADOCERO      DS 2     ; GRADO DE PERTENENCIA PARA EL ERROR CUANDO
ES CERO.
GRADOPOSNEG    DS 2     ; GRADO DE PERTENENCIA PARA EL ERROR
CUANDO ES NEGATIVO O POSITIVO.
GRADO          DS 2    ; MINIMO ENTRE LAS DOS VARIABLES DE GRADO DE
PERTENENCIA
NUM            DS 2    ; NUMERADOR DEL RESULTADO FINAL
DEN            DS 1    ; DENOMINADOR DEL RESULTADO FINAL.

$include 'MAt.RAM'

   org RomStart        ;Define el Origen de la ROM (donde quedara el programa).

$include 'LCDRTS.ASM'   ; Rutinas del LCD, Impresion de cadenas y Conversiones
Frecuentes.
$include 'SPIRTS.ASM'   ; Rutinas de Lectura, Escritura, etc. del SPI.
$include 'NUMRTS.ASM'   ; Rutinas de Multiplicacion, Division, Resta y Visualizacion.
$include 'MCPRTS.ASM'   ; Rutinas del MCP2510.
$include 'MotorRutinas.asm' ;Rutinas que manipulan el motor y mueven la valvula
*****************************************************************
* I_MENU_PPAL -
*****************************************************************
I_MENU_PPAL:
     BSET    7,INSTRUC
     LDA     #$01
     JSR     SEND
     LDA     #$03
     JSR     SEND
     BCLR    7,INSTRUC
     LDHX    #PPAL
     JSR     SEND_CAD2
     BSET    7,INSTRUC
```

```
    LDA     #$C0
    JSR     SEND
    BCLR    7,INSTRUC
    LDHX    #TNIVEL
    JSR     SEND_CAD
    BSET    7,INSTRUC
    LDA     #$CB
    JSR     SEND
    BCLR    7,INSTRUC
    LDHX    #TPOS
    JSR     SEND_CAD
    CLR     VAR
    CLRA
    JSR     HEXBCD
    lda     span
    RTS
************************************************************
* Init_Valvula: - SUBRUTINA QUE INICIALIZA LA VALVULA    *
************************************************************
Init_Valvula:
    MOV     #$FF,DDRD
    MOV     #$00,PTD
    CLR     DIGITO
    CLR     VALOR
    CLR     TECLA
    CLR     OPCION1
    CLR     OPCION2
    CLR     NIVEL
    CLR     VACTUAL
    CLR     VACTUAL_P
    CLR     VFINAL
    CLR     NVUELTAS
    CLR     kCERO
    mov     #$100T,SPAN
    MOV     #$40,KSPAN
    RTS
************************************************************************
* PRESENTACIÓN - Visualiza la primera pantalla en el LCD
************************************************************************
PRESENTACION:
    bset    7,instruc
    lda     #$98
    jsr     send
    bclr    7,instruc
    ldhx    #titulo
```

```
    jsr     send_cad
    bset    7,instruc
    lda     #$80
    jsr     send
    bclr    7,instruc
    ldhx    #titulof
    jsr     send_cad
    ldx     #$12
    jsr     desder
    bset    7,instruc
    lda     #$D6
    jsr     send
    bclr    7,instruc
    ldhx    #tit2
    jsr     send_cad
    RTS
*************************************************************
* PRINCIPAL - ESTA RUTINA SE EJECUTA DESPUES DEL RESET.    *
*                                                           *
*************************************************************
PRINCIPAL:
    rsp
    clra
    clrx
    CLR     VAR
    CLR     var1
    jsr     init_lcd
    JSR     PRESENTACION
    jsr     Init_Valvula
    jsr     INIT_SPI
    jsr     Init_MCP2510
    jsr     INI_POS
    JSR     I_MENU_PPAL
    JSR     LimpiaRam
    CLR     CONDICIONES
    CLR     NivelTant
    CLR     NivelTant+1
    CLR     NivelTant+2
    CLR     NivelTant+3
    CLR     NivelTact
    CLR     NivelTact+1
    CLR     NivelTact+2
    CLR     NivelTact+3
    CLR     GRADOCERO
    CLR     GRADOCERO+1
```

```
      CLR     GRADOPOSNEG
      CLR     GRADOPOSNEG+1
      CLR     GRADOV
      CLR     GRADOV+1
      CLR     GRADOVNC
      CLR     GRADOVNC+1
      CLR     GRADO
      CLR     GRADO+1
      CLR     NUM
      CLR     NUM+1
      CLR     DEN
      mov     #$00,Referencia
      mov     #$00,Referencia+1
      mov     #$00,Referencia+2
      mov     #$00,Referencia+3      ;Referencia en 10 Cm
      cli
ciclo
      clrx   ;Borrar el registro X
      stx     copctl  ;Cargar lo que esta en X al registro copctl (desabilita WatchDog)
      bra     ciclo  ;Repertir ciclo
*****************************************************************
* IRQ1_ISR - IRQ1 Rutina de Interrupcion para              *
*          Recepcion de datos del MCP2510.              *
*****************************************************************
IRQ1_ISR:
      pshh
      ;jsr     print
       ldx     #$62
      jsr     Read_SPI
      cbeqa    #$60,DatoReferencia
      cbeqa    #$20,DatoNivel
      bra     FinDato
DatoNivel
       BSR    EjecutarProceso
       bra    findato
DatoReferencia
      ldx     #$66
      jsr     Read_SPI
      sta     Referencia
      ldx     #$67
      jsr     Read_SPI
      sta     Referencia+1
      ldx     #$68
      jsr     Read_SPI
      sta     Referencia+2
```

```
      ldx     #$69
      jsr     Read_SPI
      sta     Referencia+3     ; En la Variable Referencia se almacenan los registros
                               ; que contienen el nivel de Referencia o deseado.
FinDato

      lda     #$00
      ldhx    #$012C
      jsr     BIT_spi      ; clarea la bandera de Buffer lleno del MCP2510
     ;jsr     print
      pulh
      rti
*********************************************************************
* EjecutarProceso - Esta Rutina Es la que se encarga de         *
*                Hacer el Control Inteligente en la          *
*                Valvula Usando para ello un algoritmo       *
*                de Contro DIFUSO.                           *
*********************************************************************
EjecutarProceso:
      ldx     #$66
      jsr     Read_SPI
      sta     NivelTact
      ldx     #$67
      jsr     Read_SPI
      sta     NivelTact+1
      ldx     #$68
      jsr     Read_SPI
      sta     NivelTact+2
      ldx     #$69
      jsr     Read_SPI
      sta     NivelTact+3
      mov     NivelTact,A
      mov     NivelTact+1,B
      mov     NivelTact+2,C
      mov     NivelTact+3,D
                     ; En la Variable NivelTact se almacenan los registros
                     ; que contienen el nivel en el tiempo ACTUAL.

      mov     NivelTant,E
      mov     NivelTant+1,F
      mov     NivelTant+2,G
      mov     NivelTant+3,H    ; En la Variable NivelTant se almacenan los registros
                     ; que contienen el nivel en el tiempo ANTERIOR.
      Jsr     Res32          ; Se resta NivelTact - NivelTant
*********************************************************************
```

```
*  AQUI SE DEFINE SI ES MAYOR MENOR O IGUAL EN CAMBIONIVEL    *
*****************************************************************
      BGE    MaEQAnt
      BSET   0,CONDICIONES    ; MAYOR QUE NIVEL ACTUAL
(DISMINUYENDO)
                        ; Nivel Actual es Menor que Nivel Anterior
      Bra    FINCMPACT
MaEQAnt
      BSET   1,CONDICIONES    ; MENOR O IGUAL QUE NIVEL ACTUAL
(AUMENTANDO)
                        ; Nivel Actual es Mayor o Igual que Nivel Anterior
FINCMPACT

      MOV    #$FF,E
      MOV    #$F0,F
      MOV    #$BD,G
      MOV    #$C0,H
      MOV    A,CambioNivel
      MOV    B,CambioNivel+1
      MOV    C,CambioNivel+2
      MOV    D,CambioNivel+3 ;GUARDA EL RESULTADO EN LA VARIABLE
CambioNivel
      JSR    RES32          ; SE RESTA EL RESULTADO ANTERIOR
                        ; EL TOPE NEGATIVO DEL "NO HAY CAMBIO".
                        ; EN ESTE CASO ES (10.00000) EN COMPLEMENTO A DOS(-
10.00000)
      BLT    NTOPE(N1)
      BSET   2,CONDICIONES    ; MAYOR IGUAL QUE TOPE -10.00000
NTOPE(N1)
      MOV    CambioNivel,A
      MOV    CambioNivel+1,B
      MOV    CambioNivel+2,C
      MOV    CambioNivel+3,D ;RESTAURA EL RESULTADO DE LA VARIABLE
CambioNivel
      MOV    #$00,E
      MOV    #$0F,F
      MOV    #$42,G
      MOV    #$40,H
      JSR    RES32          ; SE RESTA EL RESULTADO ANTERIOR
                        ; EL TOPE POSITIVO DEL "NO HAY CAMBIO".
                        ; EN ESTE CASO ES (10.00000)
      BGT    NTOPE(1)
      BSET   3,CONDICIONES   ; MENOR O IGUAL QUE TOPE 10.00000
NTOPE(1)
*****************************************************************
```

```
*  AQUI TERMINA SI ES MAYOR MENOR O IGUAL EN CAMBIONIVEL      *
*************************************************************
*************************************************************************
*****
*    Hasta aqui se ha calculado el Cambio de Nivel                        *
*************************************************************************
*****
      mov    NivelTact,E
      mov    NivelTact+1,F
      mov    NivelTact+2,G
      mov    NivelTact+3,H
      mov    Referencia,A
      mov    Referencia+1,B
      mov    Referencia+2,C
      mov    Referencia+3,D        ;REFERENCIA - NIVELTACT
      jsr    RES32
*****************************************************************
*  AQUI SE DEFINE SI ES MAYOR MENOR O IGUAL EN ERRORNIVEL     *
*****************************************************************
      BGE    MaEQNivel
      BSET   4,CONDICIONES         ; MAYOR QUE NIVEL REFERENCIA
(NEGATIVO)
                        ; Referencia es menor que nivel actual
      BRA    FINCMPREF
MaEQNivel
      BSET   5,CONDICIONES         ; MENOR O IGUAL QUE REFERENCIA
(POSITIVO)
                         ; Referencia Mayor o Igual que Nivel Actual
FINCMPREF
      MOV    #$FF,E
      MOV    #$F0,F
      MOV    #$BD,G
      MOV    #$C0,H              ;SE CARGA 1000000 EN COMPLEMENTO A DOS (-
10.00000)
      MOV    A,ErrorNivel
      MOV    B,ErrorNivel+1
      MOV    C,ErrorNivel+2
      MOV    D,ErrorNivel+3  ;GUARDA EL RESULTADO EN LA VARIABLE
ErrorNivel
      JSR    RES32         ; SE RESTA EL RESULTADO ANTERIOR
                     ; EL TOPE NEGATIVO DEL "IGUAL A REFERENCIA".
                     ; EN ESTE CASO ES (1000000) EN COMPLEMENTO A DOS(-
1000000)
      BLT    NTOPE(N10)
      BSET   6,CONDICIONES   ; MAYOR IGUAL QUE TOPE -10.00000
```

```
NTOPE(N10)
      MOV     ErrorNivel,A
      MOV     ErrorNivel+1,B
      MOV     ErrorNivel+2,C
      MOV     ErrorNivel+3,D  ;RESTAURA EL RESULTADO DE LA VARIABLE
ErrorNivel
      MOV     #$00,E
      MOV     #$0F,F
      MOV     #$42,G
      MOV     #$40,H          ;SE CARGA 1000000 QUE REPRESENTA 10.00000
      JSR     RES32           ; SE RESTA EL RESULTADO ANTERIOR
                              ; EL TOPE POSITIVO DEL "IGUAL A REFERENCIA".
                              ; EN ESTE CASO ES (10.00000)
      BGT     NTOPE(10)
      BSET    7,CONDICIONES   ; MENOR O IGUAL QUE TOPE 10.00000
NTOPE(10)
*********************************************************************
*  AQUI TERMINA DEFINIR SI ES MAYOR MENOR O IGUAL EN ERRORNIVEL  *
*********************************************************************
      Mov     NivelTact,NivelTant
      Mov     NivelTact+1,NivelTant+1
      Mov     NivelTact+2,NivelTant+2
      Mov     NivelTact+3,NivelTant+3    ; SE ACTULIZA LA VARIABLE DE NIVEL
TIEMPO ANTERIOR
*********************************************************************
*  EVALUACION DE LAS CONDICIONES Y APLICACION DE LAS ECUACIONES
*
*********************************************************************
      MOV     CambioNivel,A
      MOV     CambioNivel+1,B
      MOV     CambioNivel+2,C
      MOV     CambioNivel+3,D ;RESTAURA EL RESULTADO DE LA VARIABLE
CambioNivel
      BRSET   0,CONDICIONES,EQU1      ; Disminuyendo [-2.0 A 0]
      BRSET   1,CONDICIONES,EQU2      ; Aumentando   [0 A 2.0]
      BRA     FINEQU1
EQU1
      JSR     NEGATIVO        ;SE CAMBIA EL SIGNO
EQU2
      mov     #$100t,E
      jsr     div32
      LDHX    C
      LDA     D
      LDX     #$100T
      DIV             ;Carga el CambioNivel al Acumulador
```

```
    CLRH
    LDX    #$10T
    DIV                ;HASTA AQUI SE DIVIDE POR 100000
    LDX    #$50T               ;50 X CambioNivel
    MUL
    STX    GRADOV
    STA    GRADOV+1
FINEQU1
    MOV    CambioNivel,A
    MOV    CambioNivel+1,B
    MOV    CambioNivel+2,C
    MOV    CambioNivel+3,D ;RESTAURA EL RESULTADO DE LA VARIABLE
CambioNivel
    LDA    #$05
    AND    CONDICIONES
    CBEQA  #$05,EQU3            ; No Cambia    [-1.0 A 0]
    LDA    #$0A
    AND    CONDICIONES
    CBEQA  #$0A,EQU4            ; No Cambia    [0 A 1.0]
                        ; ECUACIONES ENTRADA CAMBIO NIVEL
    BRA    FINEQUCB
EQU3
    JSR    NEGATIVO
EQU4
    mov    #$100t,E
    jsr    div32
    LDHX   C
    LDA    D
    LDX    #$100T
    DIV                ; Carga CambioNivel en el Acumulador
    CLRH
    LDX    #$10T
    DIV                ; HASTA AQUI SE DIVIDE POR 100000
    LDX    #$100T
    MUL                ; 100 X CambioNivel
    STX    A
    STA    B
    LDA    #$E8
    SUB    B
    STA    GRADOVNC+1      ; RESTA PARCIAL 100.0 - 100 X CambioNivel
    LDA    #$03          ; 03E8 (100.0)
    SBC    A
    STA    GRADOVNC     ; RESTA FINAL 100.0 - 100 x CambioNivel
FINEQUCB
    MOV    ErrorNivel,A
```

```
      MOV     ErrorNivel+1,B
      MOV     ErrorNivel+2,C
      MOV     ErrorNivel+3,D  ;RESTAURA EL RESULTADO DE LA VARIABLE
ErrorNivel
      BRSET   4,CONDICIONES,EQU5      ; Negativo [-25.0 A 0]
      BRSET   5,CONDICIONES,EQU6      ; Positivo [0 A 25.0]
      BRA     FINEQU5
EQU5
      JSR     NEGATIVO        ; SE CAMBIA EL SIGNO
EQU6
      mov     #$100t,E
      jsr     div32
      LDHX    C
      LDA     D
      LDX     #$100T
      DIV                     ; Carga el ErrorNivel al Acumulador
      LDX     #$04T           ; 4 X ErrorNivel
      MUL
      STX     GRADOPOSNEG
      STA     GRADOPOSNEG+1
FINEQU5
      MOV     ErrorNivel,A
      MOV     ErrorNivel+1,B
      MOV     ErrorNivel+2,C
      MOV     ErrorNivel+3,D  ;RESTAURA EL RESULTADO DE LA VARIABLE
ErrorNivel
      LDA     #$50
      AND     CONDICIONES
      CBEQA   #$50,EQU7             ; Cero [-10.0 A 0]
      LDA     #$A0
      AND     CONDICIONES
      CBEQA   #$A0,EQU8             ; Cero [0 A 10.0]
                              ; ECUCAIONES DE ENTRADA ERROR NIVEL
      BRA     FINEQUER

EQU7
      JSR     NEGATIVO
EQU8
      mov     #$100t,E
      jsr     div32
      LDHX    C
      LDA     D
      LDX     #$100T
      DIV                     ; Carga el ErrorNivel al Acumulador
      LDX     #$10T
```

```
      MUL                ; 10 X ErrorNivel
      STX    A
      STA    B
      LDA    #$E8
      SUB    B
      STA    GRADOCERO+1       ; RESTA PARCIAL 100.0 - 10 X ErrorNivel
      LDA    #$03          ; 03E8 (100.0)
      SBC    A
      STA    GRADOCERO     ; RESTA FINAL 100.0 - 10 x ErrorNivel
FINEQUER

*************************************************************************
****
*   INICIO DE LA EVALUACION DE LAS REGLAS DE CONTROL DIFUSO
*
*************************************************************************
****
      LDA    #$11
      AND    CONDICIONES
      CBEQA  #$11,NEGYDIS
      BRA    FINNEGYDIS
NEGYDIS
      MOV    GRADOPOSNEG,A
      MOV    GRADOPOSNEG+1,B
      MOV    GRADOV,C
      MOV    GRADOV+1,D
      LDHX   A
      CPHX   C
      BLO    MENORQUE
      MOV    GRADOV,GRADO
      MOV    GRADOV+1,GRADO+1
      BRA    FINMENORQUE
MENORQUE
      MOV    GRADOPOSNEG,GRADO
      MOV    GRADOPOSNEG+1,GRADO+1
FINMENORQUE
      JSR    CERRAR
FINNEGYDIS
      LDA    #$C1
      AND    CONDICIONES
      CBEQA  #$C1,CEROYDIS
      BRA    FINCEROYDIS
CEROYDIS
      MOV    GRADOCERO,A
      MOV    GRADOCERO+1,B
```

```
        MOV     GRADOV,C
        MOV     GRADOV+1,D
        LDHX    A
        CPHX    C
        BLO     MENORQUE1
        MOV     GRADOV,GRADO
        MOV     GRADOV+1,GRADO+1
        BRA     FINMENORQUE1
MENORQUE1
        MOV     GRADOCERO,GRADO
        MOV     GRADOCERO+1,GRADO+1
FINMENORQUE1
        JSR     MANTENER
FINCEROYDIS
        LDA     #$21
        AND     CONDICIONES
        CBEQA   #$21,POSYDIS
        BRA     FINPOSYDIS
POSYDIS
        MOV     GRADOPOSNEG,A
        MOV     GRADOPOSNEG+1,B
        MOV     GRADOV,C
        MOV     GRADOV+1,D
        LDHX    A
        CPHX    C
        BLO     MENORQUE2
        MOV     GRADOV,GRADO
        MOV     GRADOV+1,GRADO+1
        BRA     FINMENORQUE2
MENORQUE2
        MOV     GRADOPOSNEG,GRADO
        MOV     GRADOPOSNEG+1,GRADO+1
FINMENORQUE2
        JSR     ABRIR

FINPOSYDIS
        LDA     #$1C
        AND     CONDICIONES
        CBEQA   #$1C,NEGYNCAM
        BRA     FINNEGYNCAM
NEGYNCAM
        MOV     GRADOPOSNEG,A
        MOV     GRADOPOSNEG+1,B
        MOV     GRADOVNC,C
        MOV     GRADOVNC+1,D
```

```
        LDHX    A
        CPHX    C
        BLO     MENORQUE3
        MOV     GRADOVNC,GRADO
        MOV     GRADOVNC+1,GRADO+1
        BRA     FINMENORQUE3
MENORQUE3
        MOV     GRADOPOSNEG,GRADO
        MOV     GRADOPOSNEG+1,GRADO+1
FINMENORQUE3
        JSR     CERRAR
FINNEGYNCAM
        LDA     #$CC
        AND     CONDICIONES
        CBEQA   #$CC,CEROYNCAM
        BRA     FINCEROYNCAM
CEROYNCAM
        MOV     GRADOCERO,A
        MOV     GRADOCERO+1,B
        MOV     GRADOVNC,C
        MOV     GRADOVNC+1,D
        LDHX    A
        CPHX    C
        BLO     MENORQUE4
        MOV     GRADOVNC,GRADO
        MOV     GRADOVNC+1,GRADO+1
        BRA     FINMENORQUE4
MENORQUE4
        MOV     GRADOCERO,GRADO
        MOV     GRADOCERO+1,GRADO+1
FINMENORQUE4
        JSR     MANTENER
FINCEROYNCAM
        LDA     #$2C
        AND     CONDICIONES
        CBEQA   #$2C,POSYNCAM
        BRA     FINPOSYNCAM
POSYNCAM
        MOV     GRADOPOSNEG,A
        MOV     GRADOPOSNEG+1,B
        MOV     GRADOVNC,C
        MOV     GRADOVNC+1,D
        LDHX    A
        CPHX    C
        BLO     MENORQUE5
```

```
        MOV     GRADOVNC,GRADO
        MOV     GRADOVNC+1,GRADO+1
        BRA     FINMENORQUE5
MENORQUE5
        MOV     GRADOPOSNEG,GRADO
        MOV     GRADOPOSNEG+1,GRADO+1
FINMENORQUE5
        JSR     ABRIR
FINPOSYNCAM
        LDA     #$12
        AND     CONDICIONES
        CBEQA   #$12,NEGYAUM
        BRA     FINNEGYAUM
NEGYAUM
        MOV     GRADOPOSNEG,A
        MOV     GRADOPOSNEG+1,B
        MOV     GRADOV,C
        MOV     GRADOV+1,D
        LDHX    A
        CPHX    C
        BLO     MENORQUE6
        MOV     GRADOV,GRADO
        MOV     GRADOV+1,GRADO+1
        BRA     FINMENORQUE6
MENORQUE6
        MOV     GRADOPOSNEG,GRADO
        MOV     GRADOPOSNEG+1,GRADO+1
FINMENORQUE6
        JSR     CERRAR
FINNEGYAUM
        LDA     #$C2
        AND     CONDICIONES
        CBEQA   #$C2,CEROYAUM
        BRA     FINCEROYAUM
CEROYAUM
        MOV     GRADOCERO,A
        MOV     GRADOCERO+1,B
        MOV     GRADOV,C
        MOV     GRADOV+1,D
        LDHX    A
        CPHX    C
        BLO     MENORQUE7
        MOV     GRADOV,GRADO
        MOV     GRADOV+1,GRADO+1
        BRA     FINMENORQUE7
```

```
MENORQUE7
      MOV     GRADOCERO,GRADO
      MOV     GRADOCERO+1,GRADO+1
FINMENORQUE7
      JSR     MANTENER
FINCEROYAUM
      LDA     #$22
      AND     CONDICIONES
      CBEQA   #$22,POSYAUM
      BRA     FINPOSYAUM
POSYAUM
      MOV     GRADOPOSNEG,A
      MOV     GRADOPOSNEG+1,B
      MOV     GRADOV,C
      MOV     GRADOV+1,D
      LDHX    A
      CPHX    C
      BLO     MENORQUE8
      MOV     GRADOVNC,GRADO
      MOV     GRADOVNC+1,GRADO+1
      BRA     FINMENORQUE8
MENORQUE8
      MOV     GRADOPOSNEG,GRADO
      MOV     GRADOPOSNEG+1,GRADO+1
FINMENORQUE8
      BSR     ABRIR
FINPOSYAUM
      CLR     CONDICIONES
*****************************************
*  FIN DE LA EVALUACION DE REGLAS       *
*****************************************
*****************************************
*   ACCION DE CONTROL                   *
*****************************************
     LDHX     NUM
     LDA      NUM+1
     LDX      DEN
     DIV                ; SE DIVIDE PARA HALLAR EL VALOR DE CONTROL
     ADD      VALOR
     CMP      #$100T
     BGT      TOPE
     CMP      #$00
     BLT      TOPE
     STA      VALOR
TOPE
```

```
      JSR     control
      CLR     NUM
      CLR     NUM+1
      CLR     DEN
      BSET    7,INSTRUC
      LDA     #$C0
      JSR     SEND
      BCLR    7,INSTRUC
      LDHX    #TNIVEL
      JSR     SEND_CAD
      MOV     REFERENCIA,A
      MOV     REFERENCIA+1,B
      MOV     REFERENCIA+2,C
      MOV     REFERENCIA+3,D
      MOV     #$100T,E
      JSR     DIV32
      LDHX    C
      LDA     D
      LDX     #$100T
      DIV
      MOV     #$0F,VAR
      JSR     HEXBCD
      BSET    7,INSTRUC
      LDA     #$CB
      JSR     SEND
      BCLR    7,INSTRUC
      CLR     VAR
      LDHX    #TPOS
      JSR     SEND_CAD
      LDA     VALOR
      JSR     HEXBCD
      lda     #$0
      ldx     #$31
      jsr     write_spi       ;Registro Parte alta Identificador
      lda     #$40
      ldx     #$32
      jsr     write_spi       ;Coloca el Identificador del Mensaje (Apertura)
      lda     valor
      ldx     #$36
      jsr     write_spi   ; Coloca los Datos en los campos de datos
      lda     #RTS0
      jsr     rts_spi
     RTS
*****************************************
*  FIN DE LA RUTINA EJECUTAR PROCESO    *
```

```
*****************************************
*****************************************************************
* ABRIR ; CALCULAR LA APERTURA DE LA VALVULA                    *
*****************************************************************
ABRIR:
      LDHX   GRADO
      LDA    GRADO+1
      LDX    #$10T
      DIV
      PSHA
      ADD    DEN          ; suma el denominador con ACUMULADOR
      STA    DEN          ; DEN=DEN+MINk
      PULA
      TAX                 ; SE MULTIPICA GRADO * GRADO
      MUL
      STX    A
      STA    B
      MOV    NUM,C
      MOV    NUM+1,D      ; SE VA ADICIONANDO.
      JSR    SUM16
      MOV    A,NUM
      MOV    B,NUM+1
      RTS
*****************************************************************
* MANTENER ; CALCULAR LA APERTURA DE LA VALVULA                 *
*****************************************************************
MANTENER:
      LDHX   GRADO
      LDA    GRADO+1
      LDX    #$10T
      DIV
      ADD    DEN          ;suma el denominador con ACUMULADOR
      STA    DEN
      RTS
*****************************************************************
* CERRAR ; CALCULAR LA APERTURA DE LA VALVULA                   *
*****************************************************************
CERRAR:
      LDHX   GRADO
      LDA    GRADO+1
      LDX    #$10T
      DIV
      PSHA
      ADD    DEN          ;suma el denominador con ACUMULADOR
      STA    DEN
```

```
      PULA
      TAX               ; SE MULTIPICA GRADO * GRADO
      MUL               ;
      STX    C
      STA    D
      MOV    NUM,A
      MOV    NUM+1,B
      JSR    RES16        ; SE VA DISMINUYENDO
      MOV    A,NUM
      MOV    B,NUM+1
      RTS
*****************************************************
*  dummy_isr - Interrupcion Tonta (No hace nada) *
*****************************************************
dummy_isr:
       rti
*************************************************************
* Vectors - Lista de vectores de interrupcion.          *
*         Definicion de subrutinas de Atencion.         *
*************************************************************
  org  VectorStart
      dw  dummy_isr   ; TIM1 Overflow Vector
      dw  dummy_isr   ; TIM1 Channel 1 Vector
      dw  dummy_isr   ; TIM1 Channel 0 Vector
      dw  irq1_isr       ; ~IRQ1
      dw  dummy_isr   ; SWI Vector
      dw  PRINCIPAL    ; Reset Vector
```

ANEXO B(Código Fuente del Programa en Visual Basic 6, para el monitoreo de la planta)

```
VERSION 5.00
Object = "{86CF1D34-0C5F-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCT2.OCX"
Object = "{248DD890-BB45-11CF-9ABC-0080C7E7B78D}#1.0#0"; "MSWINSCK.OCX"
Object = "{6DE878DD-9CB9-11D2-B578-CBA8FE2C1C77}#1.0#0"; "SCOPE.OCX"
Begin VB.Form FrmTanque
   Caption         =   "Sistemea Control de Nivel"
   ClientHeight    =   6480
   ClientLeft      =   615
   ClientTop       =   885
   ClientWidth     =   8430
   LinkTopic       =   "Form1"
   ScaleHeight     =   6480
   ScaleWidth      =   8430
   Begin VB.ListBox List1
      Height          =   1425
      ItemData        =   "ControldeNivel.frx":0000
      Left            =   4440
      List            =   "ControldeNivel.frx":0002
      TabIndex        =   22
      Top             =   4920
      Width           =   3615
   End
   Begin VB.Frame Frame3
      BackColor       =   &H80000007&
      Height          =   1455
      Left            =   240
      TabIndex        =   15
      Top             =   4920
      Width           =   3855
      Begin VB.Label lblHostName
         BackColor       =   &H80000007&
         Caption         =   "Host:"
         ForeColor       =   &H0000FFFF&
         Height          =   375
         Left            =   120
         TabIndex        =   21
         Top             =   240
         Width           =   615
      End
      Begin VB.Label lblIP
         BackColor       =   &H80000012&
         Caption         =   "IP Address:"
         ForeColor       =   &H0000FFFF&
         Height          =   375
         Left            =   120
         TabIndex        =   20
         Top             =   720
         Width           =   855
      End
      Begin VB.Label lblUsers
```

```
      BackColor       =   &H80000012&
      Caption         =   "Connections:"
      ForeColor       =   &H0000FFFF&
      Height          =   375
      Left            =   120
      TabIndex        =   19
      Top             =   1080
      Width           =   975
   End
   Begin VB.Label lblAddress
      BackColor       =   &H80000012&
      ForeColor       =   &H000000FF&
      Height          =   255
      Left            =   1200
      TabIndex        =   18
      Top             =   720
      Width           =   2175
   End
   Begin VB.Label lblHostID
      BackColor       =   &H80000012&
      ForeColor       =   &H000000FF&
      Height          =   255
      Left            =   1200
      TabIndex        =   17
      Top             =   360
      Width           =   2175
   End
   Begin VB.Label lblConnections
      BackColor       =   &H80000012&
      ForeColor       =   &H000000FF&
      Height          =   255
      Left            =   1200
      TabIndex        =   16
      Top             =   1080
      Width           =   2175
   End
End
Begin MSWinsockLib.Winsock Winsock1
   Left            =   0
   Top             =   4680
   _ExtentX        =   741
   _ExtentY        =   741
   _Version        =   393216
End
Begin Scope10.Scope ScpApertura
   Height          =   3135
   Left            =   4200
   TabIndex        =   14
   Top             =   120
   Width           =   3975
   _ExtentX        =   7011
   _ExtentY        =   5530
   title           =   "Apertura de la Valvula"
   xlabel          =   "Tiempo"
```

```
   ylabel          =   "%"
End
Begin Scope10.Scope ScpNivel
   Height          =   3015
   Left            =   240
   TabIndex        =   13
   Top             =   120
   Width           =   3615
   _ExtentX        =   6376
   _ExtentY        =   5318
   title           =   "Nivel de la Planta"
   xlabel          =   "Tiempo"
   ylabel          =   "Nivel"
   maxy            =   25
   nmuestras       =   1000
End
Begin VB.Timer Timer1
   Enabled         =   0   'False
   Interval        =   55
   Left            =   3960
   Top             =   4560
End
Begin VB.Frame Frame2
   Caption         =   "Medidas"
   Height          =   1452
   Left            =   4320
   TabIndex        =   1
   Top             =   3360
   Width           =   3972
   Begin VB.Label Label4
      Caption         =   "Apertura Valvula:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   -1  'True
         Strikethrough   =   0   'False
      EndProperty
      Height          =   255
      Left            =   1920
      TabIndex        =   5
      Top             =   360
      Width           =   1935
   End
   Begin VB.Label Label3
      Caption         =   "Nivel:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
```

```
         Italic          =   -1  'True
         Strikethrough   =   0   'False
      EndProperty
      Height          =   255
      Left            =   120
      TabIndex        =   4
      Top             =   360
      Width           =   1335
   End
   Begin VB.Label LblApertura
      BackColor       =   &H00FFFFFF&
      BorderStyle     =   1  'Fixed Single
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   18
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   615
      Left            =   1920
      TabIndex        =   3
      Top             =   600
      Width           =   1815
   End
   Begin VB.Label LblNivel
      BackColor       =   &H00FFFFFF&
      BorderStyle     =   1  'Fixed Single
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   18
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   615
      Left            =   120
      TabIndex        =   2
      Top             =   600
      Width           =   1575
   End
End
Begin VB.Frame Frame1
   Caption         =   "Controles"
   Height          =   1452
   Left            =   240
   TabIndex        =   0
   Top             =   3360
   Width           =   3972
   Begin VB.CommandButton Cmdsalir
```

```
   Caption         =   "Salir"
   Enabled         =   0   'False
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   12
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   492
   Left            =   1920
   TabIndex        =   10
   Top             =   840
   Width           =   1812
End
Begin VB.CommandButton CmdIniciar
   Caption         =   "Iniciar"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   12
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   492
   Left            =   1920
   TabIndex        =   9
   Top             =   240
   Width           =   1812
End
Begin VB.TextBox TxtReferencia
   DragMode        =   1  'Automatic
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   12
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   495
   Left            =   120
   TabIndex        =   7
   Top             =   480
   Width           =   1092
End
Begin MSComCtl2.UpDown UpDown1
   Height          =   495
   Left            =   1200
```

```
         TabIndex        =   6
         Top             =   480
         Width           =   240
         _ExtentX        =   450
         _ExtentY        =   873
         _Version        =   393216
         Enabled         =   -1  'True
      End
      Begin VB.Label Label7
         Caption         =   "Status:"
         Height          =   255
         Left            =   120
         TabIndex        =   11
         Top             =   1080
         Width           =   615
      End
      Begin VB.Label LblEstado
         BorderStyle     =   1  'Fixed Single
         Height          =   255
         Left            =   840
         TabIndex        =   12
         Top             =   1080
         Width           =   735
      End
      Begin VB.Label Label5
         Caption         =   "Referencia:"
         BeginProperty Font
            Name            =   "MS Sans Serif"
            Size            =   9.75
            Charset         =   0
            Weight          =   700
            Underline       =   0   'False
            Italic          =   -1  'True
            Strikethrough   =   0   'False
         EndProperty
         Height          =   252
         Left            =   120
         TabIndex        =   8
         Top             =   240
         Width           =   1332
      End
   End
End
Attribute VB_Name = "FrmTanque"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim Referencia As Double

Sub Proceso()
Dim Var As Single
Dim Rx0ID As Integer
Dim DLC As Byte
```

```
Dim RXData(0 To 7) As Double
Dim Nivel As Double
Dim Apertura As Byte



Option Explicit
Dim iSockets As Integer
Dim sServerMsg As String
Dim sRequestID As String

Private Sub Form_Load()

    Form1.Show
    lblHostID.Caption = Socket(0).LocalHostName
    lblAddress.Caption = Socket(0).LocalIP
    Socket(0).LocalPort = 1007
    sServerMsg = "Listening to port: " & Socket(0).LocalPort
    List1.AddItem (sServerMsg)
    Socket(0).Listen
End Sub

Private Sub socket_Close(Index As Integer)
    sServerMsg = "Connection closed: " & Socket(Index).RemoteHostIP
    List1.AddItem (sServerMsg)
    Socket(Index).Close
    Unload Socket(Index)
    iSockets = iSockets - 1
    lblConnections.Caption = iSockets

End Sub

Private Sub socket_ConnectionRequest(Index As Integer, ByVal requestID As
Long)
    sServerMsg = "Connection request id " & requestID & " from " &
Socket(Index).RemoteHostIP
  If Index = 0 Then
    List1.AddItem (sServerMsg)
    sRequestID = requestID
    iSockets = iSockets + 1
    lblConnections.Caption = iSockets
    Load Socket(iSockets)
    Socket(iSockets).LocalPort = 1007
    Socket(iSockets).Accept requestID
  End If

End Sub

Private Sub socket_DataArrival(Index As Integer, ByVal bytesTotal As
Long)

   Dim sItemData As String
   Dim strData As String
   Dim strOutData As String
```

```
   Dim strConnect As String


    ' get data from client
    Socket(Index).GetData sItemData, vbString
    sServerMsg = "Received: " & sItemData & " from " &
Socket(Index).RemoteHostIP & "(" & sRequestID & ")"
    List1.AddItem (sServerMsg)

    'strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=G:\Prices.mdb;Persist Security Info=False"
    Dim strPath As String

    'Change the database path in the text file

    Dim fso As New FileSystemObject, txtfile, _
    fil1 As File, ts As TextStream

    Set fil1 = fso.GetFile("path.txt")
    ' Read the contents of the file.
    Set ts = fil1.OpenAsTextStream(ForReading)
    strPath = ts.ReadLine
    ts.Close
    Set fso = Nothing

    strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Persist Security Info=False;Data Source=" & strPath & _
      "; Mode=Read|Write"

    Dim rs As New ADODB.Recordset

    ' Get clients request from database
    strData = "Item = '" & sItemData & "'"

    rs.Open "select * from prices", strConnect, adOpenKeyset,
adLockOptimistic
    rs.Find strData
    strOutData = rs.Fields("Price")

    'send data to client
    sServerMsg = "Sending: " & strOutData & " to " &
Socket(Index).RemoteHostIP
    List1.AddItem (sServerMsg)
    Socket(Index).SendData strOutData

End Sub




If Read_SPI(&H62&) And &H8& Then
    Var = 256
```

```
    Rx0ID = CStr((Read_SPI(&H61&) * 2097152) Or (((Read_SPI(&H62&) And
&HE0&) / 32) * 262144) Or ((Read_SPI(&H62&) And &H3&) * 65536) Or
((Read_SPI(&H63&)) * Var) Or (Read_SPI(&H64&)))

Else
    Rx0ID = CStr(Read_SPI(&H61&) * 8 Or (Read_SPI(&H62&) And &HE0&) / 32)

End If


Select Case Rx0ID

Case 1

        DLC = Read_SPI(&H65&)

        For i = 0 To DLC
            RXData(i) = CStr(Read_SPI(&H66& + i))
        Next


        Nivel = (((RXData(1) * 65536) + (RXData(2) * 256) + RXData(3)) /
100000)
        ScpNivel.graficar (Nivel)
        LblNivel = CStr(Nivel)
Case 2

        Apertura = CStr(Read_SPI(&H66&))
        ScpNivel.graficar (Apertura)
        LblApertura = CStr(Apertura)
Case 3


End Select

dummy = Bit_SPI(&H2C&, &H1&, 0)
dummy = Bit_SPI(&H2C&, &H2&, 0)



End Sub
Private Sub CmdIniciar_Click()
Static Boton As Boolean

If Boton Then ' Detiene Proceso y Cambia Etiqueta
   CmdIniciar.Caption = "Iniciar"

   Cmdsalir.Enabled = True
   'Habilita el Timer del sistema
   Timer1.Enabled = False
   'Escribe 0V en la DAQ
'   Status = AO_VWrite(device, aocanal, 0)
   Boton = Not Boton
   Set_Config_Mode
```

```
Else            'Inicia Proceso y Cambia etiqueta
   CmdIniciar.Caption = "Detener"
   Cmdsalir.Enabled = False
   'Habilita el Timer del sistema
   Timer1.Enabled = True

   Boton = Not Boton
   Set_Normal_Mode
End If

End Sub


Private Sub Cmdsalir_Click()
Unload FrmTanque
End Sub

Private Sub Form_Load()

'****************************************
'Iniciliza la tarjeta ISA-CAN
'****************************************

ScpNivel.inicializar
ScpApertura.inicializar
TxtReferencia = "0"
Timer1.Enabled = False
'Init_MCP2510
End Sub


Sub Init_MCP2510()

Reset_SPI
dummy = Write_SPI(&H30&, &H3&)        'TXB0CTRL
dummy = Write_SPI(&H60&, &H60&)        'TXB0CTRL
dummy = Write_SPI(&H70&, &H60&)        'RXB1CTRL
dummy = Write_SPI(&HF&, &H84&)           'CANCTRL

dummy = Write_SPI(&H35&, &H8&)   'Configura para Trama de Datos

dummy = Write_SPI(&H2A&, 1)         'CNF1
dummy = Write_SPI(&H29&, &HA0&)     'CNF2
dummy = Write_SPI(&H28&, &H2&)       'CNF3

'CmbRXB0OPM.ListIndex = 3
'CmbPrioridad.ListIndex = 3
'CmbModoOp.ListIndex = 4

'For i = 0 To 7
    'dummy = Write_SpI(54 + i, Val(TxtData(i).Text))
'Next

''TxtMask_Change
```

```
'For i = 0 To 5
  '  TxtFiltro_Change (i)
'Next
PrioridadMsg
ModoCualquier

End Sub

Private Sub EnviarReferencia()
Dim Dato As Integer

Dato = (3 And 2040) / 8          'Identificador para el Setpoit
dummy = Write_SPI(&H31&, Dato)  'Escribe en el registro TXB0SIDH
'Print Dato
Dato = (3 And 7) * 32     '(&H380000&)
dummy = Write_SPI(&H32&, Dato)  'Escribe en el registro TXB0SIDL
'General
Dato = Referencia * 10
dummy = Write_SPI(&H35&, 1)  ' Numero de Byte a Enviar (DLC).
dummy = Write_SPI(&H36&, Dato) ' Dato 0-250 de Referencia
dummy = Rts_SPI(1)

End Sub
' cambia modo de operacion al bufer de recepcion RXB0.

Sub ModoCualquier()
dummy = Bit_SPI(&H60&, &H60&, 3 * 32) ' RXB0CTRL REGISTRO DE CONTROL DEL
BUFFER DE RECEPCION CERO
' Recepcion de cualquier mensaje
End Sub


Sub PrioridadMsg()
dummy = Write_SPI(&H30&, 3) ' Prioridad de mensajes en el buffer de
transmision
End Sub


Private Sub TxtReferencia_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then
    If TxtReferencia = Char Or TxtReferencia = "" Then
        TxtReferencia = Format(Referencia, "fixed")
    Else
        If CDbl(TxtReferencia) > 25 Or CDbl(TxtReferencia) < 0 Then
            TxtReferencia = Format(Referencia, "fixed")
        Else
            'Actualiza el voltaje que esta en txtreferencia
            Referencia = CDbl(TxtReferencia)
        End If
    End If
End If
EnviarReferencia
End Sub
```

```
Private Sub Timer1_Timer()

Proceso

End Sub

Private Sub UpDown1_DownClick()

Referencia = CDbl(TxtReferencia)
If Referencia > 0 Then
    TxtReferencia = Format((Referencia - 0.5), "fixed")
Else
    TxtReferencia = Format(Referencia, "fixed")
End If
'Actualiza el voltaje que esta en txtreferencia
Referencia = CDbl(TxtReferencia)
EnviarReferencia
End Sub



Private Sub UpDown1_UpClick()

Referencia = CDbl(TxtReferencia)
If Referencia < 25 Then
    TxtReferencia = Format((Referencia + 0.5), "fixed")
Else
    TxtReferencia = Format(Referencia, "fixed")
End If
'Actualiza el voltaje que esta en txtreferencia
Referencia = CDbl(TxtReferencia)
EnviarReferencia
End Sub
```

ANEXO C (Programa en visual C++ para crear la librería de conexión dinamica para manejo del chip MCP2510 con la computadora spicmd.dll)

```
// Esta la funcion que saca datos por un puerto

void _stdcall outportb(short Dir,unsigned char Dato)
{
        _asm push ax; asm push dx; asm push cx;       // ;carga contador para enviar 8 bits. "Dirreccion del
registro a leer".

   _asm {
        mov             dx,Dir
        mov             AL,Dato
        out     dx,AL       // ;Pone el Dato en la Dir. Espesificada.
        }
        _asm pop cx; pop dx; pop ax

};

unsigned char _stdcall inportb(short Dir)
{
        unsigned char Dato;

        _asm push ax; asm push dx; asm push cx;       // ;carga contador para enviar 8 bits. "Dirreccion del
registro a leer".

   _asm {
        mov             dx,Dir
        in              AL,dx
        mov             Dato,AL
                                // ;Pone el Dato en la Dir. Espesificada.
        }
        return ( Dato );
        _asm pop cx; pop dx; pop ax
};

void  _stdcall send(unsigned char dato)
{
        _asm push ax; asm push dx; asm push cx;
        _asm mov cx,0x08;    // ;carga contador para enviar 8 bits. "Dirreccion del registro a leer".

SigBit: _asm {
        mov      dx,0x303
        rcl     dato,1       // ;Rota a la Izquierda dato.
        jc      Cuno        // salta a Cuno si Carry es uno.
        mov          AL,0x00
        out     dx,AL        // ;SCK=0 y SO=0

        mov      AL,0x04
   out     dx,AL        // ;SCK=1 y SO=0
```

```
   mov   AL,0x00
   out     dx,AL       // ;SCK=0 y SO=0
   jmp   sig
         }

Cuno:  _asm {
         mov      AL,0x02
         out dx,AL       // ;SCK=0 y SO=1
         mov      AL,0x06
         out      dx,AL       // ;SCK=1 y SO=1
         mov      AL,0x02
         out      dx,AL       // ;SCK=0 y SO=1

         }
sig:   _asm {
         loop   SigBit    // ; Decrementa el contador y salta si no es Cero.
         mov      AL,0x0
         out      dx,AL       // ;SCK=0
         pop cx; pop dx; pop ax
         }

};


unsigned char _stdcall Read_SPI(unsigned char direccion)
{
 unsigned char dato;
         outportb(0x303,0x0);

         send(0x03);//  Envia 11 (2 unos) al SPI "Instruccion de Lectura"


         send(direccion); // Envia la Direccion del registro a leer del MCP2510

  _asm {
           push ax; push dx; push cx;
         mov    cx,0x08      // ;carga contador para enviar 8 bits. "Dirreccion del registro a leer".
         }
nextbit: _asm {
         mov      dx,0x303
         mov      AL,0x04
         out      dx,AL       // ;SCK=1 y SO=0

         mov      dx,0x302
         in       AL,dx      // captura del dato SI
         and      AL,0x01    // mascara para extraer el bit 1
         jz       bcero  //  ;si el bit en SI es cero salta a bcero
         stc       //    ;Sino carry en Uno.
         jmp      sig
         }
bcero:  _asm clc     //       deja el carry en cero.
sig:   _asm rcl    dato,1  //   Rota el dato a la izquierda a traves del carry.
         _asm{
         mov      dx,0x303
```

```
        mov     AL,0x00
        out     dx,AL       // ;SCK=0 y SO=0
        loop    nextbit
        mov     dx,0x303
        mov     AL,0x01
        out     dx,AL       // ;SCK=0 y SO=0 y SC=1
        pop cx; pop dx; pop ax;
        }
        return ( dato );
};

unsigned char _stdcall ReadStatus_SPI()
{
 unsigned char dato;
        outportb(0x303,0x0);

        send(0xA0);//  Envia 10100000  al SPI "Instruccion de Lectura de Estado"

 _asm {
        push ax; push dx; push cx
        mov     cx,0x08     // ;carga contador para enviar 8 bits. "Dirreccion del registro a leer".
        }
nextbit: _asm {
        mov     dx,0x303
        mov     AL,0x04
        out     dx,AL       // ;SCK=1 y SO=0

        mov     dx,0x302
        in      AL,dx       // captura del dato SI
        and     AL,0x01     // mascara para extraer el bit 1
        jz      bcero  //  ;si el bit en SI es cero salta a bcero
        stc      //    ;Sino carry en Uno.
        jmp     sig
        }
bcero:  _asm clc     //       deja el carry en cero.
sig:    _asm {
        rcl     dato,1  //   Rota el dato a la izquierda a traves del carry.
        mov     dx,0x303
        mov     AL,0x00
        out     dx,AL       // ;SCK=0 y SO=0
        loop    nextbit
        mov     dx,0x303
        mov     AL,0x01
        out     dx,AL       // ;SCK=0 y SO=0 y SC=1
        }
        return ( dato );
}

void _stdcall Write_SPI(unsigned char direccion,unsigned char dato)
{
        outportb(0x303,0x0);

        send(0x02);   //       ; Envia 10 (1 y 0) al SPI "Instruccion de Escritura"
```

```
        send(direccion); //  ; Envia 8 bits "Dirreccion del registro a escribir".
        send(dato);//        ;enviar el dato de 8 bits.
        outportb(0x303,0x01);

};
void _stdcall Reset_SPI()
{
        outportb(0x303,0x0);

        send(0xC0);   //  ; Envia 11000000  al SPI "Instruccion de Reset"

        outportb(0x303,0x01);

};

void _stdcall Bit_SPI(unsigned char direccion,unsigned char mascara,unsigned char dato)
{
        outportb(0x303,0x0);

        send(0x05); //    ; Envia 00000101 al SPI "Instruccion de modificacion de Bit"

        send(direccion); //  ; Envia 8 bits "Dirreccion del registro a escribir".
        send(mascara);
        send(dato);//        ;enviar el dato de 8 bits.
        outportb(0x303,0x01);

};

void _stdcall Rts_SPI(unsigned char TXBn)
{
        unsigned char RTS_Valor;

        RTS_Valor = 0x80 | TXBn;
        outportb(0x303,0x0);

        send(RTS_Valor); //    ; Envia 00000101 al SPI "Instruccion de requerimiento de transmision"

        outportb(0x303,0x01);


};

void _stdcall Set_Normal_Mode()
{
        Bit_SPI(0x2B, 0x40, 0x40);
        Bit_SPI(0x2C, 0x40, 0x40);
        Bit_SPI(0x0F, 0xE0, 0x00);    //Normal mode
        Bit_SPI(0x2B, 0x40, 0x0);
        Bit_SPI(0x2C, 0x40, 0x0);

};
void _stdcall Set_Sleep_Mode()
{
        Bit_SPI(0x0F, 0xE0, 0x20);    //Sleep mode
```

```
};


void _stdcall Set_Listen_Mode()
{
        Bit_SPI(0x2B, 0x40, 0x40);
        Bit_SPI(0x2C, 0x40, 0x40);

        if (!(Read_SPI(0x0E) & 0xE0))
                Bit_SPI(0x0F, 0xE0, 0x40);    //LoopBack mode

        Bit_SPI(0x0F, 0xE0, 0x60);    //Listen mode
        Bit_SPI(0x2B, 0x40, 0x0);
        Bit_SPI(0x2C, 0x40, 0x0);

};

void _stdcall Set_LoopBack_Mode()
{
        Bit_SPI(0x2B, 0x40, 0x40);
        Bit_SPI(0x2C, 0x40, 0x40);
        Bit_SPI(0x0F, 0xE0, 0x40);    //LoopBack mode
        Bit_SPI(0x2B, 0x40, 0x0);
        Bit_SPI(0x2C, 0x40, 0x0);

};
void _stdcall Set_Config_Mode()
{
        Bit_SPI(0x2B, 0x40, 0x40);
        Bit_SPI(0x2C, 0x40, 0x40);
        Bit_SPI(0x0F, 0xE0, 0x80);    //Config. mode/
        Bit_SPI(0x2B, 0x40, 0x0);
        Bit_SPI(0x2C, 0x40, 0x0);

};
```